Appendix A

Appendix A illustrates semantic types that may be supported and their corresponding adaptive template names. For example, the Pipelined semantic type is made up of, in this order, the map_keys the pipe_state and the index_fact adaptive templates. The example pre-parsed and post parsed SQL adaptive templates are then provided.

As mentioned previously, the use of the semantic types significantly reduces the amount of work needed to implement the datamart 150. By selecting a semantic type for a particular fact table or dimension table, the consultant automatically selects the corresponding pre-parsed SQL adaptive templates. The selected adaptive templates are then automatically converted into post parsed SQL statements that include the schema specific information for the datamart 150. Additionally, these post parsed SQL statements include the SQL for accessing and manipulating the datamart 150 tables.

semantic_type_name	adaptive_template_name
Pipelined	map_keys
Pipelined	pipe_state
Pipelined	index_fact
Pipelined/Unjoined	upd_unj
Pipelined/Unjoined	map_keys
Pipelined/Unjoined	pipe_state
Pipelined/Unjoined	index_fact
Slowly Changing Dimensions	insert_dim
Slowly Changing Dimensions	index_dim
Transactional	map_keys
Transactional	load_trans
Transactional	ren_trans
Transactional	index_fact
Transactional/Inventory	map_keys

Transactional/Inventory	load_trans
Transactional/Inventory	inv_adjust
Transactional/Inventory	index_fact
Transactional/Inventory/ForceZero	map_keys
Transactional/Inventory/ForceZero	load_trans
Transactional/Inventory/ForceZero	force_zero
Transactional/Inventory/ForceZero	inv_adjust
Transactional/Inventory/ForceZero	index_fact
Transactional/Inventory/ForceZero/Unjoined	upd_unj
Transactional/Inventory/ForceZero/Unjoined	map_keys
Transactional/Inventory/ForceZero/Unjoined	load_trans
Transactional/Inventory/ForceZero/Unjoined	force_zero
Transactional/Inventory/ForceZero/Unjoined	inv_adjust
Transactional/Inventory/ForceZero/Unjoined	index_fact
Transactional/Inventory/Unjoined	upd_unj
Transactional/Inventory/Unjoined	map_keys
Transactional/Inventory/Unjoined	load_trans
Transactional/Inventory/Unjoined	inv_adjust
Transactional/Inventory/Unjoined	index_fact
Transactional/Statelike	map_keys
Transactional/Statelike	load_trans
Transactional/Statelike	load_state
Transactional/Statelike	index_fact
Transactional/Statelike/ForceClose	map_keys
Transactional/Statelike/ForceClose	load_trans
Transactional/Statelike/ForceClose	force_close
Transactional/Statelike/ForceClose	load_state
Transactional/Statelike/ForceClose	index_fact
Transactional/Statelike/ForceClose/Unjoined	upd_unj
Transactional/Statelike/ForceClose/Unjoined	map_keys
Transactional/Statelike/ForceClose/Unjoined	load_trans
Transactional/Statelike/ForceClose/Unjoined	force_close
Transactional/Statelike/ForceClose/Unjoined	load_state

Transactional/Statelike/ForceClose/Unjoined	index_fact
Transactional/Statelike/Unjoined	upd_unj
Transactional/Statelike/Unjoined	map_keys
Transactional/Statelike/Unjoined	load_trans
Transactional/Statelike/Unjoined	load_state
Transactional/Statelike/Unjoined	index_fact
Transactional/Unjoined	upd_unj
Transactional/Unjoined	map_keys
Transactional/Unjoined	load_trans
Transactional/Unjoined	ren_trans
Transactional/Unjoined	index_fact

The following are the pre-parsed pseudo-SQL source for the adaptive templates.

```
--#TEMPLATE_BEGIN# force_close

/*****************************************************************************/
--
-- Copyright * 1997, Epiphany Marketing Software, Inc.  All Rights Reserved.
--
-- force_close
--
-- Close out deleted orders - those that no longer appear in the
-- staging table
--
-- SEE SAFETY VALVE BELOW
--
/*****************************************************************************/

/*****************************************************************************/
-- Delete temporary tables
/*****************************************************************************/

--#BLOCK_BEGIN# DropTemps

$$DDL_BEGIN
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_FC]
$$DDL_END

--#BLOCK_END# DropTemps

/*****************************************************************************/
-- Insert negative BOOKs for deleted orders
--
-- FC: ForceClose
/*****************************************************************************/

--#BLOCK_BEGIN# MakeFC

$$SELECT_INTO_BEGIN[$$FCTTBL[]_FC]
SELECT
        f.iss,
        f.ss_key,
        MAX(f.date_key) date_key,
        MIN(f.transtype_key) transtype_key,
```

```
        MAX(f.seq) + 1 seq
,       f.$$DIMKEYR_01
,       f.$$DIMKEYR_02
,       f.$$DIMKEYR_03
,       f.$$DIMKEYR_04
,       f.$$DIMKEYR_05
,       f.$$DIMKEYR_06
,       f.$$DIMKEYR_07
,       f.$$DIMKEYR_08
,       f.$$DIMKEYR_09
,       f.$$DIMKEYR_10
,       f.$$DEGKEY_01
,       f.$$DEGKEY_02
,       f.$$DEGKEY_03

,       -SUM(f.$$FCTCOL_001) $$FCTCOL_001
,       -SUM(f.$$FCTCOL_002) $$FCTCOL_002
,       -SUM(f.$$FCTCOL_003) $$FCTCOL_003
,       -SUM(f.$$FCTCOL_004) $$FCTCOL_004
,       -SUM(f.$$FCTCOL_005) $$FCTCOL_005
,       -SUM(f.$$FCTCOL_006) $$FCTCOL_006
,       -SUM(f.$$FCTCOL_007) $$FCTCOL_007
,       -SUM(f.$$FCTCOL_008) $$FCTCOL_008
,       -SUM(f.$$FCTCOL_009) $$FCTCOL_009
,       -SUM(f.$$FCTCOL_010) $$FCTCOL_010
,       -SUM(f.$$FCTCOL_011) $$FCTCOL_011
,       -SUM(f.$$FCTCOL_012) $$FCTCOL_012
,       -SUM(f.$$FCTCOL_013) $$FCTCOL_013
,       -SUM(f.$$FCTCOL_014) $$FCTCOL_014
,       -SUM(f.$$FCTCOL_015) $$FCTCOL_015
,       -SUM(f.$$FCTCOL_016) $$FCTCOL_016
,       -SUM(f.$$FCTCOL_017) $$FCTCOL_017
,       -SUM(f.$$FCTCOL_018) $$FCTCOL_018
,       -SUM(f.$$FCTCOL_019) $$FCTCOL_019
,       -SUM(f.$$FCTCOL_020) $$FCTCOL_020
,       -SUM(f.$$FCTCOL_021) $$FCTCOL_021
,       -SUM(f.$$FCTCOL_022) $$FCTCOL_022
,       -SUM(f.$$FCTCOL_023) $$FCTCOL_023
,       -SUM(f.$$FCTCOL_024) $$FCTCOL_024

$$SELECT_INTO_BODY[$$FCTTBL[]_FC]
FROM
        $$FCTTBL[]$$CURR f
WHERE
        NOT EXISTS
        (SELECT 1 FROM $$FSTGTBL[]_MAP s WHERE s.iss = f.iss AND s.ss_key = f.ss_key)
GROUP BY
        f.iss,
        f.ss_key
,       f.$$DIMKEYR_01
,       f.$$DIMKEYR_02
,       f.$$DIMKEYR_03
,       f.$$DIMKEYR_04
,       f.$$DIMKEYR_05
,       f.$$DIMKEYR_06
,       f.$$DIMKEYR_07
,       f.$$DIMKEYR_08
,       f.$$DIMKEYR_09
,       f.$$DIMKEYR_10
,       f.$$DEGKEY_01
,       f.$$DEGKEY_02
,       f.$$DEGKEY_03

HAVING
        (
        (SUM(f.$$FCTCOL_001) <> 0)
 OR     (SUM(f.$$FCTCOL_002) <> 0)
 OR     (SUM(f.$$FCTCOL_003) <> 0)
 OR     (SUM(f.$$FCTCOL_004) <> 0)
 OR     (SUM(f.$$FCTCOL_005) <> 0)
 OR     (SUM(f.$$FCTCOL_006) <> 0)
```

```
 OR     (SUM(f.$$FCTCOL_007) <> 0)
 OR     (SUM(f.$$FCTCOL_008) <> 0)
 OR     (SUM(f.$$FCTCOL_009) <> 0)
 OR     (SUM(f.$$FCTCOL_010) <> 0)
 OR     (SUM(f.$$FCTCOL_011) <> 0)
 OR     (SUM(f.$$FCTCOL_012) <> 0)
 OR     (SUM(f.$$FCTCOL_013) <> 0)
 OR     (SUM(f.$$FCTCOL_014) <> 0)
 OR     (SUM(f.$$FCTCOL_015) <> 0)
 OR     (SUM(f.$$FCTCOL_016) <> 0)
 OR     (SUM(f.$$FCTCOL_017) <> 0)
 OR     (SUM(f.$$FCTCOL_018) <> 0)
 OR     (SUM(f.$$FCTCOL_019) <> 0)
 OR     (SUM(f.$$FCTCOL_020) <> 0)
 OR     (SUM(f.$$FCTCOL_021) <> 0)
 OR     (SUM(f.$$FCTCOL_022) <> 0)
 OR     (SUM(f.$$FCTCOL_023) <> 0)
 OR     (SUM(f.$$FCTCOL_024) <> 0)
        )
AND
        MIN(f.transtype_key) <= 99
AND
        MIN(f.transtype_key) >= 1

--#BLOCK_END# MakeFC

/*****************************************************************************/
-- SAFETY VALVE - THIS PROC ONLY DOES ANYTHING
-- IF THE STAGING TABLE HAS AT LEAST ONE ROW
/*****************************************************************************/

--#BLOCK_BEGIN# SafetyValue

DECLARE $$VAR[count_MAP] $$EPIINT$$EOS

BEGIN

$$VAR_ASSIGN_BEGIN[count_MAP]
SELECT COUNT(1)
$$VAR_ASSIGN_INTO[count_MAP]
FROM $$FSTGTBL[]_MAP
$$VAR_ASSIGN_END

$$IF[($$VAR[count_MAP] = 0)]
DELETE FROM $$FCTTBL[]_FC$$EOS
$$END_IF

END$$EOS

--#BLOCK_END# SafetyValue

/*****************************************************************************/
-- Count processed, inserted rows
/*****************************************************************************/

--#BLOCK_BEGIN# SPResults

BEGIN

INSERT INTO adaptive_template_profile (token_name, number_rows)
SELECT 'PROCESSED', COUNT(1) FROM $$FCTTBL[]$$CURR$$EOS

INSERT INTO adaptive_template_profile (token_name, number_rows)
SELECT 'INSERTED', COUNT(1) FROM $$FCTTBL[]_FC$$EOS

END$$EOS

--#BLOCK_END# SPResults

--#TEMPLATE_END# force_close
```

```
--#TEMPLATE_BEGIN# load_state

/*****************************************************************************/
--
-- Copyright * 1997, Epiphany Marketing Software, Inc.  All Rights Reserved.
--
-- load_state
--
-- Load order bookings into fact table by creating transactional
-- data from state data
--
-- load_trans must be run before this procedure to create TIN table
--
/*****************************************************************************/

/*****************************************************************************/
-- Delete temporary tables
/*****************************************************************************/

--#BLOCK_BEGIN# DropTemps

$$DDL_BEGIN
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_MFL]
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_1ST]
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_IL]
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_IR]
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_IRD]
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_IND]
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_NFD]
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_IRM]
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_IDM]
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_ILM]
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_IMI]
$$DDL_END

--#BLOCK_END# DropTemps

/*********************************************************************/
-- Set join order for SQL Server
/*********************************************************************/

--#BLOCK_BEGIN# ForcePlanOn

$$SQLSERVER[SET FORCEPLAN ON]

--#BLOCK_END# ForcePlanOn

/*****************************************************************************/
-- Remove rows older than fact table - history can not be rewritten - only
-- the last date for an order can be changed. Note that we compare transtype's
-- because SHIP type transactions might occur at a later date and we don't want
-- those to interfere
--
-- Also, since the staging table may have multiple entries for a given order on
-- a single day - we assume that the list one inserted in the Staging table will
-- be used (since ikey is an IDENTITY column)
--
-- Note that a given ss_key must use the same Booking transtype for all of time,
-- otherwise the transtype_key
--
-- MFL: Mapped Filtered
/*****************************************************************************/

--#BLOCK_BEGIN# MakeMFL

$$SELECT_INTO_BEGIN[$$FCTTBL[]_MFL]
SELECT
        s.*
$$SELECT_INTO_BODY[$$FCTTBL[]_MFL]
FROM
```

```
        $$FSTGTBL[]_MAP s, bus_process b
WHERE
        ((s.date_key >= (SELECT MAX(date_key) FROM $$FCTTBL[]$$CURR f WHERE
                s.iss = f.iss AND s.ss_key = f.ss_key AND
                s.transtype_key = f.transtype_key))
        OR NOT EXISTS (SELECT * FROM $$FCTTBL[]$$CURR f WHERE
                s.iss = f.iss AND s.ss_key = f.ss_key AND
                s.transtype_key = f.transtype_key))
AND     s.ikey = (SELECT MAX(t.ikey) FROM $$FSTGTBL[]_MAP t WHERE
                s.iss = t.iss AND
                s.ss_key = t.ss_key AND
                s.date_key = t.date_key AND
                t.process_key = b.process_key)
AND
        s.process_key = b.process_key AND b.process_name = 'LoadState'

--#BLOCK_END# MakeMFL

/*****************************************************************************/
-- Index MFL table for later queries
/*****************************************************************************/

--#BLOCK_BEGIN# IndexMFL

$$DDL_BEGIN
$$DDL_EXEC[
CREATE INDEX X$$FCTTBL[]_MFL ON $$FCTTBL[]_MFL
(
 iss, ss_key, date_key
)
]
$$DDL_END

--#BLOCK_END# IndexMFL

/*****************************************************************************/
-- Get oldest state rows for each unique sskey
--
-- We need to treat the first entry for each order
-- in the staging table separately from all others, since
-- only the first entry needs to be compared with
-- already existing fact entry rows to create transactions.
-- All subsequent dates for that order in the Fact table
-- can be delta'd with other staging table entries - see the
-- section below on Pairwise deltas.
--
-- MFL should be indexed
--
-- 1ST: The first record for each iss, ss_key
/*****************************************************************************/

--#BLOCK_BEGIN# Make1ST

$$SELECT_INTO_BEGIN[$$FCTTBL[]_1ST]
SELECT
        s.*
$$SELECT_INTO_BODY[$$FCTTBL[]_1ST]
FROM
        $$FCTTBL[]_MFL s
WHERE
        s.date_key = (SELECT MIN(date_key) FROM $$FCTTBL[]_MFL t WHERE
                s.iss = t.iss AND s.ss_key = t.ss_key)

--#BLOCK_END# Make1ST

/*****************************************************************************/
-- Index 1ST for later queries
/*****************************************************************************/

--#BLOCK_BEGIN# Index1ST
```

```
$$DDL_BEGIN
$$DDL_EXEC[
CREATE UNIQUE INDEX XPK$$FCTTBL[]_1ST ON $$FCTTBL[]_1ST
(
 iss, ss_key
)
]
$$DDL_END

--#BLOCK_END# Index1ST

/*****************************************************************************/
-- Insert negative BOOKs for changed dim keys
--
-- This query will add up all existing Books and Loss's
-- for this order and the net facts will be cancelled out
-- with the old Dimension keys. Note that an invariant of this
-- procedure is that only one set of dimensions at a time
-- can have non-zero facts.
--
-- Fact table Should be indexed
--
-- HAVING Clause is needed to prevent changing of dimensions
-- on fully shipped order from causing a transaction - no sense
-- creating fact rows with all zero's in them
--
-- Note that we increment the sequence number just in case
-- this new transaction occurs on the same date as the last
-- existing one in the fact table - to avoid index errors
--
-- IL: InsertLost
/*****************************************************************************/

--#BLOCK_BEGIN# MakeIL

$$SELECT_INTO_BEGIN[$$FCTTBL[]_IL]
SELECT
       s.iss,
       s.ss_key,
       s.date_key,
       s.transtype_key,
       MAX(f.seq) + 1 seq
,      f.$$DIMKEYR_01
,      f.$$DIMKEYR_02
,      f.$$DIMKEYR_03
,      f.$$DIMKEYR_04
,      f.$$DIMKEYR_05
,      f.$$DIMKEYR_06
,      f.$$DIMKEYR_07
,      f.$$DIMKEYR_08
,      f.$$DIMKEYR_09
,      f.$$DIMKEYR_10
,      f.$$DEGKEY_01
,      f.$$DEGKEY_02
,      f.$$DEGKEY_03

,      -SUM(f.$$FCTCOL_001) $$FCTCOL_001
,      -SUM(f.$$FCTCOL_002) $$FCTCOL_002
,      -SUM(f.$$FCTCOL_003) $$FCTCOL_003
,      -SUM(f.$$FCTCOL_004) $$FCTCOL_004
,      -SUM(f.$$FCTCOL_005) $$FCTCOL_005
,      -SUM(f.$$FCTCOL_006) $$FCTCOL_006
,      -SUM(f.$$FCTCOL_007) $$FCTCOL_007
,      -SUM(f.$$FCTCOL_008) $$FCTCOL_008
,      -SUM(f.$$FCTCOL_009) $$FCTCOL_009
,      -SUM(f.$$FCTCOL_010) $$FCTCOL_010
,      -SUM(f.$$FCTCOL_011) $$FCTCOL_011
,      -SUM(f.$$FCTCOL_012) $$FCTCOL_012
,      -SUM(f.$$FCTCOL_013) $$FCTCOL_013
,      -SUM(f.$$FCTCOL_014) $$FCTCOL_014
,      -SUM(f.$$FCTCOL_015) $$FCTCOL_015
```

```
,       -SUM(f.$$FCTCOL_016) $$FCTCOL_016
,       -SUM(f.$$FCTCOL_017) $$FCTCOL_017
,       -SUM(f.$$FCTCOL_018) $$FCTCOL_018
,       -SUM(f.$$FCTCOL_019) $$FCTCOL_019
,       -SUM(f.$$FCTCOL_020) $$FCTCOL_020
,       -SUM(f.$$FCTCOL_021) $$FCTCOL_021
,       -SUM(f.$$FCTCOL_022) $$FCTCOL_022
,       -SUM(f.$$FCTCOL_023) $$FCTCOL_023
,       -SUM(f.$$FCTCOL_024) $$FCTCOL_024

$$SELECT_INTO_BODY[$$FCTTBL[]_IL]
FROM
        $$FCTTBL[]_1ST s, $$FCTTBL[]$$CURR f
WHERE
        s.iss = f.iss AND s.ss_key = f.ss_key
AND
        ((s.$$DIMKEYR_06 <> f.$$DIMKEYR_06) OR
        (s.$$DIMKEYR_05 <> f.$$DIMKEYR_05) OR
        (s.$$DIMKEYR_07 <> f.$$DIMKEYR_07) OR
        (s.$$DIMKEYR_04 <> f.$$DIMKEYR_04) OR
        (s.$$DIMKEYR_08 <> f.$$DIMKEYR_08) OR
        (s.$$DIMKEYR_03 <> f.$$DIMKEYR_03) OR
        (s.$$DIMKEYR_09 <> f.$$DIMKEYR_09) OR
        (s.$$DIMKEYR_02 <> f.$$DIMKEYR_02) OR
        (s.$$DIMKEYR_10 <> f.$$DIMKEYR_10) OR
        (s.$$DIMKEYR_01 <> f.$$DIMKEYR_01) )
GROUP BY
        s.iss,
        s.ss_key,
        s.date_key,
        s.transtype_key
,       f.$$DIMKEYR_01
,       f.$$DIMKEYR_02
,       f.$$DIMKEYR_03
,       f.$$DIMKEYR_04
,       f.$$DIMKEYR_05
,       f.$$DIMKEYR_06
,       f.$$DIMKEYR_07
,       f.$$DIMKEYR_08
,       f.$$DIMKEYR_09
,       f.$$DIMKEYR_10
,       f.$$DEGKEY_01
,       f.$$DEGKEY_02
,       f.$$DEGKEY_03

HAVING
        MIN(f.transtype_key) = s.transtype_key
AND
        (
        (SUM(f.$$FCTCOL_001) <> 0)
OR      (SUM(f.$$FCTCOL_002) <> 0)
OR      (SUM(f.$$FCTCOL_003) <> 0)
OR      (SUM(f.$$FCTCOL_004) <> 0)
OR      (SUM(f.$$FCTCOL_005) <> 0)
OR      (SUM(f.$$FCTCOL_006) <> 0)
OR      (SUM(f.$$FCTCOL_007) <> 0)
OR      (SUM(f.$$FCTCOL_008) <> 0)
OR      (SUM(f.$$FCTCOL_009) <> 0)
OR      (SUM(f.$$FCTCOL_010) <> 0)
OR      (SUM(f.$$FCTCOL_011) <> 0)
OR      (SUM(f.$$FCTCOL_012) <> 0)
OR      (SUM(f.$$FCTCOL_013) <> 0)
OR      (SUM(f.$$FCTCOL_014) <> 0)
OR      (SUM(f.$$FCTCOL_015) <> 0)
OR      (SUM(f.$$FCTCOL_016) <> 0)
OR      (SUM(f.$$FCTCOL_017) <> 0)
OR      (SUM(f.$$FCTCOL_018) <> 0)
OR      (SUM(f.$$FCTCOL_019) <> 0)
OR      (SUM(f.$$FCTCOL_020) <> 0)
OR      (SUM(f.$$FCTCOL_021) <> 0)
OR      (SUM(f.$$FCTCOL_022) <> 0)
```

```
OR      (SUM(f.$$FCTCOL_023) <> 0)
OR      (SUM(f.$$FCTCOL_024) <> 0)
        )

--#BLOCK_END# MakeIL

/*****************************************************************************/
-- Index IL for later queries
/*****************************************************************************/

--#BLOCK_BEGIN# IndexIL

$$DDL_BEGIN
$$DDL_EXEC[
CREATE INDEX XPK$$FCTTBL[]_IL ON $$FCTTBL[]_IL
(
 iss, ss_key
)
]
$$DDL_END

--#BLOCK_END# IndexIL

/*****************************************************************************/
-- Insert BOOKs for changed dim keys
--
-- When a dimension changes then just create a booking
-- transaction for whatever we negated above with the new
-- dimension and fact values
--
-- 1ST shoud be indexed
--
-- Note that we add one to whatever we used as the last
-- seq because this transaction occurs on the same
-- date as the negative one above
--
-- IR: Insert Rebook
/*****************************************************************************/

--#BLOCK_BEGIN# MakeIR

$$SELECT_INTO_BEGIN[$$FCTTBL[]_IR]
SELECT
        s.iss,
        s.ss_key,
        s.date_key,
        l.transtype_key,
        l.seq + 1 seq
,       s.$$DIMKEYR_01
,       s.$$DIMKEYR_02
,       s.$$DIMKEYR_03
,       s.$$DIMKEYR_04
,       s.$$DIMKEYR_05
,       s.$$DIMKEYR_06
,       s.$$DIMKEYR_07
,       s.$$DIMKEYR_08
,       s.$$DIMKEYR_09
,       s.$$DIMKEYR_10
,       s.$$DEGKEY_01
,       s.$$DEGKEY_02
,       s.$$DEGKEY_03

,       -1.$$FCTCOL_001 $$FCTCOL_001
,       -1.$$FCTCOL_002 $$FCTCOL_002
,       -1.$$FCTCOL_003 $$FCTCOL_003
,       -1.$$FCTCOL_004 $$FCTCOL_004
,       -1.$$FCTCOL_005 $$FCTCOL_005
,       -1.$$FCTCOL_006 $$FCTCOL_006
,       -1.$$FCTCOL_007 $$FCTCOL_007
,       -1.$$FCTCOL_008 $$FCTCOL_008
,       -1.$$FCTCOL_009 $$FCTCOL_009
```

```
,       -1.$$FCTCOL_010 $$FCTCOL_010
,       -1.$$FCTCOL_011 $$FCTCOL_011
,       -1.$$FCTCOL_012 $$FCTCOL_012
,       -1.$$FCTCOL_013 $$FCTCOL_013
,       -1.$$FCTCOL_014 $$FCTCOL_014
,       -1.$$FCTCOL_015 $$FCTCOL_015
,       -1.$$FCTCOL_016 $$FCTCOL_016
,       -1.$$FCTCOL_017 $$FCTCOL_017
,       -1.$$FCTCOL_018 $$FCTCOL_018
,       -1.$$FCTCOL_019 $$FCTCOL_019
,       -1.$$FCTCOL_020 $$FCTCOL_020
,       -1.$$FCTCOL_021 $$FCTCOL_021
,       -1.$$FCTCOL_022 $$FCTCOL_022
,       -1.$$FCTCOL_023 $$FCTCOL_023
,       -1.$$FCTCOL_024 $$FCTCOL_024

$$SELECT_INTO_BODY[$$FCTTBL[]_IR]
FROM
        $$FCTTBL[]_IL l, $$FCTTBL[]_1ST s
WHERE l.iss = s.iss AND l.ss_key = s.ss_key

--#BLOCK_END# MakeIR

/*****************************************************************************/
-- Insert BOOKs for changed dim keys where fact
-- also changed
--
-- When a dimension changes at the same time as
-- a fact then we need to make up the fact difference
--
-- 1ST shoud be indexed
--
-- Note that we add two to whatever we used as the last
-- seq because this transaction occurs on the same
-- date as the negative and positive ones above
--
-- Note also that the Left Outer join uses transtype_key
-- so that only the Bookings at the old value will be counted.
-- Whereas above for the negative transaction value
-- we want to include Shipments in our calculation, here
-- we only want to see how Booking Facts have changed.
--
-- Here again, only one Booking transaction type is supported
-- per ss_key
--
-- IRD: Insert Rebook delta
/*****************************************************************************/

--#BLOCK_BEGIN# MakeIRD

$$SELECT_INTO_BEGIN[$$FCTTBL[]_IRD]
SELECT
        s.iss,
        s.ss_key,
        s.date_key,
        s.transtype_key,
        l.seq + 2 seq
,       s.$$DIMKEYR_01
,       s.$$DIMKEYR_02
,       s.$$DIMKEYR_03
,       s.$$DIMKEYR_04
,       s.$$DIMKEYR_05
,       s.$$DIMKEYR_06
,       s.$$DIMKEYR_07
,       s.$$DIMKEYR_08
,       s.$$DIMKEYR_09
,       s.$$DIMKEYR_10
,       s.$$DEGKEY_01
,       s.$$DEGKEY_02
,       s.$$DEGKEY_03
```

```
,       MAX(s.$$FCTCOL_001)-$$NVL[SUM(f.$$FCTCOL_001) ~,~ 0] $$FCTCOL_001
,       MAX(s.$$FCTCOL_002)-$$NVL[SUM(f.$$FCTCOL_002) ~,~ 0] $$FCTCOL_002
,       MAX(s.$$FCTCOL_003)-$$NVL[SUM(f.$$FCTCOL_003) ~,~ 0] $$FCTCOL_003
,       MAX(s.$$FCTCOL_004)-$$NVL[SUM(f.$$FCTCOL_004) ~,~ 0] $$FCTCOL_004
,       MAX(s.$$FCTCOL_005)-$$NVL[SUM(f.$$FCTCOL_005) ~,~ 0] $$FCTCOL_005
,       MAX(s.$$FCTCOL_006)-$$NVL[SUM(f.$$FCTCOL_006) ~,~ 0] $$FCTCOL_006
,       MAX(s.$$FCTCOL_007)-$$NVL[SUM(f.$$FCTCOL_007) ~,~ 0] $$FCTCOL_007
,       MAX(s.$$FCTCOL_008)-$$NVL[SUM(f.$$FCTCOL_008) ~,~ 0] $$FCTCOL_008
,       MAX(s.$$FCTCOL_009)-$$NVL[SUM(f.$$FCTCOL_009) ~,~ 0] $$FCTCOL_009
,       MAX(s.$$FCTCOL_010)-$$NVL[SUM(f.$$FCTCOL_010) ~,~ 0] $$FCTCOL_010
,       MAX(s.$$FCTCOL_011)-$$NVL[SUM(f.$$FCTCOL_011) ~,~ 0] $$FCTCOL_011
,       MAX(s.$$FCTCOL_012)-$$NVL[SUM(f.$$FCTCOL_012) ~,~ 0] $$FCTCOL_012
,       MAX(s.$$FCTCOL_013)-$$NVL[SUM(f.$$FCTCOL_013) ~,~ 0] $$FCTCOL_013
,       MAX(s.$$FCTCOL_014)-$$NVL[SUM(f.$$FCTCOL_014) ~,~ 0] $$FCTCOL_014
,       MAX(s.$$FCTCOL_015)-$$NVL[SUM(f.$$FCTCOL_015) ~,~ 0] $$FCTCOL_015
,       MAX(s.$$FCTCOL_016)-$$NVL[SUM(f.$$FCTCOL_016) ~,~ 0] $$FCTCOL_016
,       MAX(s.$$FCTCOL_017)-$$NVL[SUM(f.$$FCTCOL_017) ~,~ 0] $$FCTCOL_017
,       MAX(s.$$FCTCOL_018)-$$NVL[SUM(f.$$FCTCOL_018) ~,~ 0] $$FCTCOL_018
,       MAX(s.$$FCTCOL_019)-$$NVL[SUM(f.$$FCTCOL_019) ~,~ 0] $$FCTCOL_019
,       MAX(s.$$FCTCOL_020)-$$NVL[SUM(f.$$FCTCOL_020) ~,~ 0] $$FCTCOL_020
,       MAX(s.$$FCTCOL_021)-$$NVL[SUM(f.$$FCTCOL_021) ~,~ 0] $$FCTCOL_021
,       MAX(s.$$FCTCOL_022)-$$NVL[SUM(f.$$FCTCOL_022) ~,~ 0] $$FCTCOL_022
,       MAX(s.$$FCTCOL_023)-$$NVL[SUM(f.$$FCTCOL_023) ~,~ 0] $$FCTCOL_023
,       MAX(s.$$FCTCOL_024)-$$NVL[SUM(f.$$FCTCOL_024) ~,~ 0] $$FCTCOL_024

$$SELECT_INTO_BODY[$$FCTTBL[]_IRD]
FROM
        $$FCTTBL[]_IL l, $$FCTTBL[]_1ST s
        $$LOJ_FROM[$$FCTTBL[]$$CURR f ~,~ s.iss = f.iss AND s.ss_key = f.ss_key AND
s.transtype_key = f.transtype_key]
WHERE
        l.iss = s.iss AND l.ss_key = s.ss_key
$$JOIN_WHERE[s.iss = f.iss (+) AND s.ss_key = f.ss_key (+) AND s.transtype_key =
f.transtype_key (+)]
GROUP BY
        s.iss,
        s.ss_key,
        s.date_key,
        s.transtype_key,
        l.seq
,       s.$$DIMKEYR_01
,       s.$$DIMKEYR_02
,       s.$$DIMKEYR_03
,       s.$$DIMKEYR_04
,       s.$$DIMKEYR_05
,       s.$$DIMKEYR_06
,       s.$$DIMKEYR_07
,       s.$$DIMKEYR_08
,       s.$$DIMKEYR_09
,       s.$$DIMKEYR_10
,       s.$$DEGKEY_01
,       s.$$DEGKEY_02
,       s.$$DEGKEY_03

HAVING
        ($$NVL[SUM(f.$$FCTCOL_001) ~,~ 0] <> MAX(s.$$FCTCOL_001))
OR      ($$NVL[SUM(f.$$FCTCOL_002) ~,~ 0] <> MAX(s.$$FCTCOL_002))
OR      ($$NVL[SUM(f.$$FCTCOL_003) ~,~ 0] <> MAX(s.$$FCTCOL_003))
OR      ($$NVL[SUM(f.$$FCTCOL_004) ~,~ 0] <> MAX(s.$$FCTCOL_004))
OR      ($$NVL[SUM(f.$$FCTCOL_005) ~,~ 0] <> MAX(s.$$FCTCOL_005))
OR      ($$NVL[SUM(f.$$FCTCOL_006) ~,~ 0] <> MAX(s.$$FCTCOL_006))
OR      ($$NVL[SUM(f.$$FCTCOL_007) ~,~ 0] <> MAX(s.$$FCTCOL_007))
OR      ($$NVL[SUM(f.$$FCTCOL_008) ~,~ 0] <> MAX(s.$$FCTCOL_008))
OR      ($$NVL[SUM(f.$$FCTCOL_009) ~,~ 0] <> MAX(s.$$FCTCOL_009))
OR      ($$NVL[SUM(f.$$FCTCOL_010) ~,~ 0] <> MAX(s.$$FCTCOL_010))
OR      ($$NVL[SUM(f.$$FCTCOL_011) ~,~ 0] <> MAX(s.$$FCTCOL_011))
OR      ($$NVL[SUM(f.$$FCTCOL_012) ~,~ 0] <> MAX(s.$$FCTCOL_012))
OR      ($$NVL[SUM(f.$$FCTCOL_013) ~,~ 0] <> MAX(s.$$FCTCOL_013))
OR      ($$NVL[SUM(f.$$FCTCOL_014) ~,~ 0] <> MAX(s.$$FCTCOL_014))
OR      ($$NVL[SUM(f.$$FCTCOL_015) ~,~ 0] <> MAX(s.$$FCTCOL_015))
OR      ($$NVL[SUM(f.$$FCTCOL_016) ~,~ 0] <> MAX(s.$$FCTCOL_016))
```

```
OR      ($$NVL[SUM(f.$$FCTCOL_017) ~,~ 0] <> MAX(s.$$FCTCOL_017))
OR      ($$NVL[SUM(f.$$FCTCOL_018) ~,~ 0] <> MAX(s.$$FCTCOL_018))
OR      ($$NVL[SUM(f.$$FCTCOL_019) ~,~ 0] <> MAX(s.$$FCTCOL_019))
OR      ($$NVL[SUM(f.$$FCTCOL_020) ~,~ 0] <> MAX(s.$$FCTCOL_020))
OR      ($$NVL[SUM(f.$$FCTCOL_021) ~,~ 0] <> MAX(s.$$FCTCOL_021))
OR      ($$NVL[SUM(f.$$FCTCOL_022) ~,~ 0] <> MAX(s.$$FCTCOL_022))
OR      ($$NVL[SUM(f.$$FCTCOL_023) ~,~ 0] <> MAX(s.$$FCTCOL_023))
OR      ($$NVL[SUM(f.$$FCTCOL_024) ~,~ 0] <> MAX(s.$$FCTCOL_024))

--#BLOCK_END# MakeIRD

/*******************************************************************************/
-- Insert BOOKs for deltas with same dim keys OR for
-- brand new orders.
--
-- Note that we DON'T want to count Shipments
-- (so shipment ss_key's should be different from
-- order ss_keys) since we just want bookings to sum up
-- to whatever this transcation says they should be.
--
-- Fact table should be indexed
--
-- WHERE clause prevents double booking on changed
-- dimension - if we didn't use the NOT EXISTS clause
-- then this query would repeat the work of the last one
-- above - which we have already taken care of
--
-- HAVING clause ensures that multiple 0 records don't
-- get inserted whenever this procedure is run
--
-- Note that we increment the sequence number just in case
-- this new transaction occurs on the same date as the last
-- existing one in the fact table - to avoid index errors
--
-- IND: Insert New Delta
/*******************************************************************************/

--#BLOCK_BEGIN# MakeIND

$$SELECT_INTO_BEGIN[$$FCTTBL[]_IND]
SELECT
        s.iss,
        s.ss_key,
        s.date_key,
        s.transtype_key,
        $$NVL[MAX(f.seq) ~,~ 0] + 1 seq
,       s.$$DIMKEYR_01
,       s.$$DIMKEYR_02
,       s.$$DIMKEYR_03
,       s.$$DIMKEYR_04
,       s.$$DIMKEYR_05
,       s.$$DIMKEYR_06
,       s.$$DIMKEYR_07
,       s.$$DIMKEYR_08
,       s.$$DIMKEYR_09
,       s.$$DIMKEYR_10
,       s.$$DEGKEY_01
,       s.$$DEGKEY_02
,       s.$$DEGKEY_03

,       MAX(s.$$FCTCOL_001)-$$NVL[SUM(f.$$FCTCOL_001) ~,~ 0] $$FCTCOL_001
,       MAX(s.$$FCTCOL_002)-$$NVL[SUM(f.$$FCTCOL_002) ~,~ 0] $$FCTCOL_002
,       MAX(s.$$FCTCOL_003)-$$NVL[SUM(f.$$FCTCOL_003) ~,~ 0] $$FCTCOL_003
,       MAX(s.$$FCTCOL_004)-$$NVL[SUM(f.$$FCTCOL_004) ~,~ 0] $$FCTCOL_004
,       MAX(s.$$FCTCOL_005)-$$NVL[SUM(f.$$FCTCOL_005) ~,~ 0] $$FCTCOL_005
,       MAX(s.$$FCTCOL_006)-$$NVL[SUM(f.$$FCTCOL_006) ~,~ 0] $$FCTCOL_006
,       MAX(s.$$FCTCOL_007)-$$NVL[SUM(f.$$FCTCOL_007) ~,~ 0] $$FCTCOL_007
,       MAX(s.$$FCTCOL_008)-$$NVL[SUM(f.$$FCTCOL_008) ~,~ 0] $$FCTCOL_008
,       MAX(s.$$FCTCOL_009)-$$NVL[SUM(f.$$FCTCOL_009) ~,~ 0] $$FCTCOL_009
,       MAX(s.$$FCTCOL_010)-$$NVL[SUM(f.$$FCTCOL_010) ~,~ 0] $$FCTCOL_010
,       MAX(s.$$FCTCOL_011)-$$NVL[SUM(f.$$FCTCOL_011) ~,~ 0] $$FCTCOL_011
```

```
,       MAX(s.$$FCTCOL_012)-$$NVL[SUM(f.$$FCTCOL_012) ~,~ 0] $$FCTCOL_012
,       MAX(s.$$FCTCOL_013)-$$NVL[SUM(f.$$FCTCOL_013) ~,~ 0] $$FCTCOL_013
,       MAX(s.$$FCTCOL_014)-$$NVL[SUM(f.$$FCTCOL_014) ~,~ 0] $$FCTCOL_014
,       MAX(s.$$FCTCOL_015)-$$NVL[SUM(f.$$FCTCOL_015) ~,~ 0] $$FCTCOL_015
,       MAX(s.$$FCTCOL_016)-$$NVL[SUM(f.$$FCTCOL_016) ~,~ 0] $$FCTCOL_016
,       MAX(s.$$FCTCOL_017)-$$NVL[SUM(f.$$FCTCOL_017) ~,~ 0] $$FCTCOL_017
,       MAX(s.$$FCTCOL_018)-$$NVL[SUM(f.$$FCTCOL_018) ~,~ 0] $$FCTCOL_018
,       MAX(s.$$FCTCOL_019)-$$NVL[SUM(f.$$FCTCOL_019) ~,~ 0] $$FCTCOL_019
,       MAX(s.$$FCTCOL_020)-$$NVL[SUM(f.$$FCTCOL_020) ~,~ 0] $$FCTCOL_020
,       MAX(s.$$FCTCOL_021)-$$NVL[SUM(f.$$FCTCOL_021) ~,~ 0] $$FCTCOL_021
,       MAX(s.$$FCTCOL_022)-$$NVL[SUM(f.$$FCTCOL_022) ~,~ 0] $$FCTCOL_022
,       MAX(s.$$FCTCOL_023)-$$NVL[SUM(f.$$FCTCOL_023) ~,~ 0] $$FCTCOL_023
,       MAX(s.$$FCTCOL_024)-$$NVL[SUM(f.$$FCTCOL_024) ~,~ 0] $$FCTCOL_024

$$SELECT_INTO_BODY[$$FCTTBL[]_IND]
FROM
        $$FCTTBL[]_1ST s $$LOJ_FROM[$$FCTTBL[]$$CURR f ~,~
                s.iss = f.iss AND s.ss_key = f.ss_key AND f.transtype_key = s.transtype_key]
WHERE
        NOT EXISTS (SELECT * FROM $$FCTTBL[]_IL WHERE iss = s.iss AND ss_key = s.ss_key)
$$JOIN_WHERE[s.iss = f.iss (+) AND s.ss_key = f.ss_key (+) AND s.transtype_key =
f.transtype_key (+)]
GROUP BY
        s.iss,
        s.ss_key,
        s.date_key,
        s.transtype_key
,       s.$$DIMKEYR_01
,       s.$$DIMKEYR_02
,       s.$$DIMKEYR_03
,       s.$$DIMKEYR_04
,       s.$$DIMKEYR_05
,       s.$$DIMKEYR_06
,       s.$$DIMKEYR_07
,       s.$$DIMKEYR_08
,       s.$$DIMKEYR_09
,       s.$$DIMKEYR_10
,       s.$$DEGKEY_01
,       s.$$DEGKEY_02
,       s.$$DEGKEY_03

HAVING
        ($$NVL[SUM(f.$$FCTCOL_001) ~,~ 0] <> MAX(s.$$FCTCOL_001))
 OR     ($$NVL[SUM(f.$$FCTCOL_002) ~,~ 0] <> MAX(s.$$FCTCOL_002))
 OR     ($$NVL[SUM(f.$$FCTCOL_003) ~,~ 0] <> MAX(s.$$FCTCOL_003))
 OR     ($$NVL[SUM(f.$$FCTCOL_004) ~,~ 0] <> MAX(s.$$FCTCOL_004))
 OR     ($$NVL[SUM(f.$$FCTCOL_005) ~,~ 0] <> MAX(s.$$FCTCOL_005))
 OR     ($$NVL[SUM(f.$$FCTCOL_006) ~,~ 0] <> MAX(s.$$FCTCOL_006))
 OR     ($$NVL[SUM(f.$$FCTCOL_007) ~,~ 0] <> MAX(s.$$FCTCOL_007))
 OR     ($$NVL[SUM(f.$$FCTCOL_008) ~,~ 0] <> MAX(s.$$FCTCOL_008))
 OR     ($$NVL[SUM(f.$$FCTCOL_009) ~,~ 0] <> MAX(s.$$FCTCOL_009))
 OR     ($$NVL[SUM(f.$$FCTCOL_010) ~,~ 0] <> MAX(s.$$FCTCOL_010))
 OR     ($$NVL[SUM(f.$$FCTCOL_011) ~,~ 0] <> MAX(s.$$FCTCOL_011))
 OR     ($$NVL[SUM(f.$$FCTCOL_012) ~,~ 0] <> MAX(s.$$FCTCOL_012))
 OR     ($$NVL[SUM(f.$$FCTCOL_013) ~,~ 0] <> MAX(s.$$FCTCOL_013))
 OR     ($$NVL[SUM(f.$$FCTCOL_014) ~,~ 0] <> MAX(s.$$FCTCOL_014))
 OR     ($$NVL[SUM(f.$$FCTCOL_015) ~,~ 0] <> MAX(s.$$FCTCOL_015))
 OR     ($$NVL[SUM(f.$$FCTCOL_016) ~,~ 0] <> MAX(s.$$FCTCOL_016))
 OR     ($$NVL[SUM(f.$$FCTCOL_017) ~,~ 0] <> MAX(s.$$FCTCOL_017))
 OR     ($$NVL[SUM(f.$$FCTCOL_018) ~,~ 0] <> MAX(s.$$FCTCOL_018))
 OR     ($$NVL[SUM(f.$$FCTCOL_019) ~,~ 0] <> MAX(s.$$FCTCOL_019))
 OR     ($$NVL[SUM(f.$$FCTCOL_020) ~,~ 0] <> MAX(s.$$FCTCOL_020))
 OR     ($$NVL[SUM(f.$$FCTCOL_021) ~,~ 0] <> MAX(s.$$FCTCOL_021))
 OR     ($$NVL[SUM(f.$$FCTCOL_022) ~,~ 0] <> MAX(s.$$FCTCOL_022))
 OR     ($$NVL[SUM(f.$$FCTCOL_023) ~,~ 0] <> MAX(s.$$FCTCOL_023))
 OR     ($$NVL[SUM(f.$$FCTCOL_024) ~,~ 0] <> MAX(s.$$FCTCOL_024))


--#BLOCK_END# MakeIND

/*****************************************************************************/
```

```
-- Form pairwise deltas for all rows except earliest for each sskey
--
-- Each row created in NFD will consist of two sequential entries from the
-- staing table. So if N enties for an order exist in MFL (after we have filtered
-- out same-date duplicates) then all the queries above will deal with the earliest entry,
whereas
-- all the queries below (including this one) will deal with the N-1 deltaing transactions
--
-- This query assumes that MFL will already have been filtered
-- to have a single record for each order/datekey
--
-- NFD: Not First Delta
/*******************************************************************************/

--#BLOCK_BEGIN# MakeNFD

$$SELECT_INTO_BEGIN[$$FCTTBL[]_NFD]
SELECT
        s.iss siss, t.iss tiss
,       s.ss_key sss_key, t.ss_key tss_key
,       s.date_key sdate_key, t.date_key tdate_key
,       s.transtype_key stranstype_key, t.transtype_key ttranstype_key
,       s.$$DIMKEYR_01 s$$DIMKEYR_01, t.$$DIMKEYR_01 t$$DIMKEYR_01
,       s.$$DIMKEYR_02 s$$DIMKEYR_02, t.$$DIMKEYR_02 t$$DIMKEYR_02
,       s.$$DIMKEYR_03 s$$DIMKEYR_03, t.$$DIMKEYR_03 t$$DIMKEYR_03
,       s.$$DIMKEYR_04 s$$DIMKEYR_04, t.$$DIMKEYR_04 t$$DIMKEYR_04
,       s.$$DIMKEYR_05 s$$DIMKEYR_05, t.$$DIMKEYR_05 t$$DIMKEYR_05
,       s.$$DIMKEYR_06 s$$DIMKEYR_06, t.$$DIMKEYR_06 t$$DIMKEYR_06
,       s.$$DIMKEYR_07 s$$DIMKEYR_07, t.$$DIMKEYR_07 t$$DIMKEYR_07
,       s.$$DIMKEYR_08 s$$DIMKEYR_08, t.$$DIMKEYR_08 t$$DIMKEYR_08
,       s.$$DIMKEYR_09 s$$DIMKEYR_09, t.$$DIMKEYR_09 t$$DIMKEYR_09
,       s.$$DIMKEYR_10 s$$DIMKEYR_10, t.$$DIMKEYR_10 t$$DIMKEYR_10
,       s.$$DEGKEY_01 s$$DEGKEY_01, t.$$DEGKEY_01 t$$DEGKEY_01
,       s.$$DEGKEY_02 s$$DEGKEY_02, t.$$DEGKEY_02 t$$DEGKEY_02
,       s.$$DEGKEY_03 s$$DEGKEY_03, t.$$DEGKEY_03 t$$DEGKEY_03
,       s.$$FCTCOL_001 s$$FCTCOL_001, t.$$FCTCOL_001 t$$FCTCOL_001
,       s.$$FCTCOL_002 s$$FCTCOL_002, t.$$FCTCOL_002 t$$FCTCOL_002
,       s.$$FCTCOL_003 s$$FCTCOL_003, t.$$FCTCOL_003 t$$FCTCOL_003
,       s.$$FCTCOL_004 s$$FCTCOL_004, t.$$FCTCOL_004 t$$FCTCOL_004
,       s.$$FCTCOL_005 s$$FCTCOL_005, t.$$FCTCOL_005 t$$FCTCOL_005
,       s.$$FCTCOL_006 s$$FCTCOL_006, t.$$FCTCOL_006 t$$FCTCOL_006
,       s.$$FCTCOL_007 s$$FCTCOL_007, t.$$FCTCOL_007 t$$FCTCOL_007
,       s.$$FCTCOL_008 s$$FCTCOL_008, t.$$FCTCOL_008 t$$FCTCOL_008
,       s.$$FCTCOL_009 s$$FCTCOL_009, t.$$FCTCOL_009 t$$FCTCOL_009
,       s.$$FCTCOL_010 s$$FCTCOL_010, t.$$FCTCOL_010 t$$FCTCOL_010
,       s.$$FCTCOL_011 s$$FCTCOL_011, t.$$FCTCOL_011 t$$FCTCOL_011
,       s.$$FCTCOL_012 s$$FCTCOL_012, t.$$FCTCOL_012 t$$FCTCOL_012
,       s.$$FCTCOL_013 s$$FCTCOL_013, t.$$FCTCOL_013 t$$FCTCOL_013
,       s.$$FCTCOL_014 s$$FCTCOL_014, t.$$FCTCOL_014 t$$FCTCOL_014
,       s.$$FCTCOL_015 s$$FCTCOL_015, t.$$FCTCOL_015 t$$FCTCOL_015
,       s.$$FCTCOL_016 s$$FCTCOL_016, t.$$FCTCOL_016 t$$FCTCOL_016
,       s.$$FCTCOL_017 s$$FCTCOL_017, t.$$FCTCOL_017 t$$FCTCOL_017
,       s.$$FCTCOL_018 s$$FCTCOL_018, t.$$FCTCOL_018 t$$FCTCOL_018
,       s.$$FCTCOL_019 s$$FCTCOL_019, t.$$FCTCOL_019 t$$FCTCOL_019
,       s.$$FCTCOL_020 s$$FCTCOL_020, t.$$FCTCOL_020 t$$FCTCOL_020
,       s.$$FCTCOL_021 s$$FCTCOL_021, t.$$FCTCOL_021 t$$FCTCOL_021
,       s.$$FCTCOL_022 s$$FCTCOL_022, t.$$FCTCOL_022 t$$FCTCOL_022
,       s.$$FCTCOL_023 s$$FCTCOL_023, t.$$FCTCOL_023 t$$FCTCOL_023
,       s.$$FCTCOL_024 s$$FCTCOL_024, t.$$FCTCOL_024 t$$FCTCOL_024

$$SELECT_INTO_BODY[$$FCTTBL[]_NFD]
FROM
        $$FCTTBL[]_MFL s, $$FCTTBL[]_MFL t
WHERE
        s.iss = t.iss AND s.ss_key = t.ss_key
AND
        s.date_key = (SELECT MAX(date_key) FROM $$FCTTBL[]_MFL u WHERE
        u.iss = s.iss AND u.ss_key = s.ss_key AND u.date_key < t.date_key)

--#BLOCK_END# MakeNFD
```

```
/*****************************************************************************/
--
-- Insert BOOKs for deltas with same dim keys
--
-- If the dimensions don't change then we create a
-- new booking order (as long as at least one of the facts
-- have changed)
--
-- IDM: Insert Delta More
--
/*****************************************************************************/

--#BLOCK_BEGIN# MakeIDM

$$SELECT_INTO_BEGIN[$$FCTTBL[]_IDM]
SELECT
        tiss iss,
        tss_key ss_key,
        tdate_key date_key,
        ttranstype_key transtype_key,
        0 seq
,       t$$DIMKEYR_01 $$DIMKEYR_01
,       t$$DIMKEYR_02 $$DIMKEYR_02
,       t$$DIMKEYR_03 $$DIMKEYR_03
,       t$$DIMKEYR_04 $$DIMKEYR_04
,       t$$DIMKEYR_05 $$DIMKEYR_05
,       t$$DIMKEYR_06 $$DIMKEYR_06
,       t$$DIMKEYR_07 $$DIMKEYR_07
,       t$$DIMKEYR_08 $$DIMKEYR_08
,       t$$DIMKEYR_09 $$DIMKEYR_09
,       t$$DIMKEYR_10 $$DIMKEYR_10
,       t$$DEGKEY_01 $$DEGKEY_01
,       t$$DEGKEY_02 $$DEGKEY_02
,       t$$DEGKEY_03 $$DEGKEY_03

,       t$$FCTCOL_001-s$$FCTCOL_001 $$FCTCOL_001
 ,      t$$FCTCOL_002-s$$FCTCOL_002 $$FCTCOL_002
 ,      t$$FCTCOL_003-s$$FCTCOL_003 $$FCTCOL_003
 ,      t$$FCTCOL_004-s$$FCTCOL_004 $$FCTCOL_004
 ,      t$$FCTCOL_005-s$$FCTCOL_005 $$FCTCOL_005
 ,      t$$FCTCOL_006-s$$FCTCOL_006 $$FCTCOL_006
 ,      t$$FCTCOL_007-s$$FCTCOL_007 $$FCTCOL_007
 ,      t$$FCTCOL_008-s$$FCTCOL_008 $$FCTCOL_008
 ,      t$$FCTCOL_009-s$$FCTCOL_009 $$FCTCOL_009
 ,      t$$FCTCOL_010-s$$FCTCOL_010 $$FCTCOL_010
 ,      t$$FCTCOL_011-s$$FCTCOL_011 $$FCTCOL_011
 ,      t$$FCTCOL_012-s$$FCTCOL_012 $$FCTCOL_012
 ,      t$$FCTCOL_013-s$$FCTCOL_013 $$FCTCOL_013
 ,      t$$FCTCOL_014-s$$FCTCOL_014 $$FCTCOL_014
 ,      t$$FCTCOL_015-s$$FCTCOL_015 $$FCTCOL_015
 ,      t$$FCTCOL_016-s$$FCTCOL_016 $$FCTCOL_016
 ,      t$$FCTCOL_017-s$$FCTCOL_017 $$FCTCOL_017
 ,      t$$FCTCOL_018-s$$FCTCOL_018 $$FCTCOL_018
 ,      t$$FCTCOL_019-s$$FCTCOL_019 $$FCTCOL_019
 ,      t$$FCTCOL_020-s$$FCTCOL_020 $$FCTCOL_020
 ,      t$$FCTCOL_021-s$$FCTCOL_021 $$FCTCOL_021
 ,      t$$FCTCOL_022-s$$FCTCOL_022 $$FCTCOL_022
 ,      t$$FCTCOL_023-s$$FCTCOL_023 $$FCTCOL_023
 ,      t$$FCTCOL_024-s$$FCTCOL_024 $$FCTCOL_024

$$SELECT_INTO_BODY[$$FCTTBL[]_IDM]
FROM
        $$FCTTBL[]_NFD d
WHERE
        (
        (s$$DIMKEYR_06 = t$$DIMKEYR_06) AND
        (s$$DIMKEYR_05 = t$$DIMKEYR_05) AND
        (s$$DIMKEYR_07 = t$$DIMKEYR_07) AND
        (s$$DIMKEYR_04 = t$$DIMKEYR_04) AND
        (s$$DIMKEYR_08 = t$$DIMKEYR_08) AND
        (s$$DIMKEYR_03 = t$$DIMKEYR_03) AND
```

```
        (s$$DIMKEYR_09 = t$$DIMKEYR_09) AND
        (s$$DIMKEYR_02 = t$$DIMKEYR_02) AND
        (s$$DIMKEYR_10 = t$$DIMKEYR_10) AND
        (s$$DIMKEYR_01 = t$$DIMKEYR_01)
        )
AND
        (
        (s$$FCTCOL_001 <> t$$FCTCOL_001)
OR      (s$$FCTCOL_002 <> t$$FCTCOL_002)
OR      (s$$FCTCOL_003 <> t$$FCTCOL_003)
OR      (s$$FCTCOL_004 <> t$$FCTCOL_004)
OR      (s$$FCTCOL_005 <> t$$FCTCOL_005)
OR      (s$$FCTCOL_006 <> t$$FCTCOL_006)
OR      (s$$FCTCOL_007 <> t$$FCTCOL_007)
OR      (s$$FCTCOL_008 <> t$$FCTCOL_008)
OR      (s$$FCTCOL_009 <> t$$FCTCOL_009)
OR      (s$$FCTCOL_010 <> t$$FCTCOL_010)
OR      (s$$FCTCOL_011 <> t$$FCTCOL_011)
OR      (s$$FCTCOL_012 <> t$$FCTCOL_012)
OR      (s$$FCTCOL_013 <> t$$FCTCOL_013)
OR      (s$$FCTCOL_014 <> t$$FCTCOL_014)
OR      (s$$FCTCOL_015 <> t$$FCTCOL_015)
OR      (s$$FCTCOL_016 <> t$$FCTCOL_016)
OR      (s$$FCTCOL_017 <> t$$FCTCOL_017)
OR      (s$$FCTCOL_018 <> t$$FCTCOL_018)
OR      (s$$FCTCOL_019 <> t$$FCTCOL_019)
OR      (s$$FCTCOL_020 <> t$$FCTCOL_020)
OR      (s$$FCTCOL_021 <> t$$FCTCOL_021)
OR      (s$$FCTCOL_022 <> t$$FCTCOL_022)
OR      (s$$FCTCOL_023 <> t$$FCTCOL_023)
OR      (s$$FCTCOL_024 <> t$$FCTCOL_024)
        )

--#BLOCK_END# MakeIDM

/*****************************************************************************/
--
-- Insert negative BOOKs for deltas with different dim keys
--
-- If one of the dimensions change then we first create a lose transaction for
-- all the previous facts. (Negate all the facts from the earlier of the two
-- transactions)
--
-- ILM: Insert Lost More
--
/*****************************************************************************/

--#BLOCK_BEGIN# MakeILM

$$SELECT_INTO_BEGIN[$$FCTTBL[]_ILM]
SELECT
        siss iss,
        sss_key ss_key,
        tdate_key date_key,
        stranstype_key transtype_key,
        0 seq
,       s$$DIMKEYR_01 $$DIMKEYR_01
,       s$$DIMKEYR_02 $$DIMKEYR_02
,       s$$DIMKEYR_03 $$DIMKEYR_03
,       s$$DIMKEYR_04 $$DIMKEYR_04
,       s$$DIMKEYR_05 $$DIMKEYR_05
,       s$$DIMKEYR_06 $$DIMKEYR_06
,       s$$DIMKEYR_07 $$DIMKEYR_07
,       s$$DIMKEYR_08 $$DIMKEYR_08
,       s$$DIMKEYR_09 $$DIMKEYR_09
,       s$$DIMKEYR_10 $$DIMKEYR_10
,       s$$DEGKEY_01 $$DEGKEY_01
,       s$$DEGKEY_02 $$DEGKEY_02
,       s$$DEGKEY_03 $$DEGKEY_03

,       -s$$FCTCOL_001 $$FCTCOL_001
```

```
,       -s$$FCTCOL_002 $$FCTCOL_002
,       -s$$FCTCOL_003 $$FCTCOL_003
,       -s$$FCTCOL_004 $$FCTCOL_004
,       -s$$FCTCOL_005 $$FCTCOL_005
,       -s$$FCTCOL_006 $$FCTCOL_006
,       -s$$FCTCOL_007 $$FCTCOL_007
,       -s$$FCTCOL_008 $$FCTCOL_008
,       -s$$FCTCOL_009 $$FCTCOL_009
,       -s$$FCTCOL_010 $$FCTCOL_010
,       -s$$FCTCOL_011 $$FCTCOL_011
,       -s$$FCTCOL_012 $$FCTCOL_012
,       -s$$FCTCOL_013 $$FCTCOL_013
,       -s$$FCTCOL_014 $$FCTCOL_014
,       -s$$FCTCOL_015 $$FCTCOL_015
,       -s$$FCTCOL_016 $$FCTCOL_016
,       -s$$FCTCOL_017 $$FCTCOL_017
,       -s$$FCTCOL_018 $$FCTCOL_018
,       -s$$FCTCOL_019 $$FCTCOL_019
,       -s$$FCTCOL_020 $$FCTCOL_020
,       -s$$FCTCOL_021 $$FCTCOL_021
,       -s$$FCTCOL_022 $$FCTCOL_022
,       -s$$FCTCOL_023 $$FCTCOL_023
,       -s$$FCTCOL_024 $$FCTCOL_024

$$SELECT_INTO_BODY[$$FCTTBL[]_ILM]
FROM
        $$FCTTBL[]_NFD d
WHERE
        (
        (s$$DIMKEYR_06 <> t$$DIMKEYR_06) OR
        (s$$DIMKEYR_05 <> t$$DIMKEYR_05) OR
        (s$$DIMKEYR_07 <> t$$DIMKEYR_07) OR
        (s$$DIMKEYR_04 <> t$$DIMKEYR_04) OR
        (s$$DIMKEYR_08 <> t$$DIMKEYR_08) OR
        (s$$DIMKEYR_03 <> t$$DIMKEYR_03) OR
        (s$$DIMKEYR_09 <> t$$DIMKEYR_09) OR
        (s$$DIMKEYR_02 <> t$$DIMKEYR_02) OR
        (s$$DIMKEYR_10 <> t$$DIMKEYR_10) OR
        (s$$DIMKEYR_01 <> t$$DIMKEYR_01)
        )
AND
        (
        (s$$FCTCOL_001 <> 0)
OR      (s$$FCTCOL_002 <> 0)
OR      (s$$FCTCOL_003 <> 0)
OR      (s$$FCTCOL_004 <> 0)
OR      (s$$FCTCOL_005 <> 0)
OR      (s$$FCTCOL_006 <> 0)
OR      (s$$FCTCOL_007 <> 0)
OR      (s$$FCTCOL_008 <> 0)
OR      (s$$FCTCOL_009 <> 0)
OR      (s$$FCTCOL_010 <> 0)
OR      (s$$FCTCOL_011 <> 0)
OR      (s$$FCTCOL_012 <> 0)
OR      (s$$FCTCOL_013 <> 0)
OR      (s$$FCTCOL_014 <> 0)
OR      (s$$FCTCOL_015 <> 0)
OR      (s$$FCTCOL_016 <> 0)
OR      (s$$FCTCOL_017 <> 0)
OR      (s$$FCTCOL_018 <> 0)
OR      (s$$FCTCOL_019 <> 0)
OR      (s$$FCTCOL_020 <> 0)
OR      (s$$FCTCOL_021 <> 0)
OR      (s$$FCTCOL_022 <> 0)
OR      (s$$FCTCOL_023 <> 0)
OR      (s$$FCTCOL_024 <> 0)
        )

--#BLOCK_END# MakeILM

/*****************************************************************************/
```

```
--
-- Insert BOOKs for deltas with different dim keys
--
-- When a dimension key changes then we can simply insert all the new facts with the
-- new dimension keys
--
-- Note that seq = 1 here because this is the second transaction on this date for
-- this order.
--
-- IRM: Insert Rebook More
--
/*****************************************************************************/

--#BLOCK_BEGIN# MakeIRM

$$SELECT_INTO_BEGIN[$$FCTTBL[]_IRM]
SELECT
        tiss iss,
        tss_key ss_key,
        tdate_key date_key,
        ttranstype_key transtype_key,
        1 seq
,       t$$DIMKEYR_01 $$DIMKEYR_01
,       t$$DIMKEYR_02 $$DIMKEYR_02
,       t$$DIMKEYR_03 $$DIMKEYR_03
,       t$$DIMKEYR_04 $$DIMKEYR_04
,       t$$DIMKEYR_05 $$DIMKEYR_05
,       t$$DIMKEYR_06 $$DIMKEYR_06
,       t$$DIMKEYR_07 $$DIMKEYR_07
,       t$$DIMKEYR_08 $$DIMKEYR_08
,       t$$DIMKEYR_09 $$DIMKEYR_09
,       t$$DIMKEYR_10 $$DIMKEYR_10
,       t$$DEGKEY_01 $$DEGKEY_01
,       t$$DEGKEY_02 $$DEGKEY_02
,       t$$DEGKEY_03 $$DEGKEY_03

,       t$$FCTCOL_001 $$FCTCOL_001
,       t$$FCTCOL_002 $$FCTCOL_002
,       t$$FCTCOL_003 $$FCTCOL_003
,       t$$FCTCOL_004 $$FCTCOL_004
,       t$$FCTCOL_005 $$FCTCOL_005
,       t$$FCTCOL_006 $$FCTCOL_006
,       t$$FCTCOL_007 $$FCTCOL_007
,       t$$FCTCOL_008 $$FCTCOL_008
,       t$$FCTCOL_009 $$FCTCOL_009
,       t$$FCTCOL_010 $$FCTCOL_010
,       t$$FCTCOL_011 $$FCTCOL_011
,       t$$FCTCOL_012 $$FCTCOL_012
,       t$$FCTCOL_013 $$FCTCOL_013
,       t$$FCTCOL_014 $$FCTCOL_014
,       t$$FCTCOL_015 $$FCTCOL_015
,       t$$FCTCOL_016 $$FCTCOL_016
,       t$$FCTCOL_017 $$FCTCOL_017
,       t$$FCTCOL_018 $$FCTCOL_018
,       t$$FCTCOL_019 $$FCTCOL_019
,       t$$FCTCOL_020 $$FCTCOL_020
,       t$$FCTCOL_021 $$FCTCOL_021
,       t$$FCTCOL_022 $$FCTCOL_022
,       t$$FCTCOL_023 $$FCTCOL_023
,       t$$FCTCOL_024 $$FCTCOL_024

$$SELECT_INTO_BODY[$$FCTTBL[]_IRM]
FROM
        $$FCTTBL[]_NFD d
WHERE
        (
        (s$$DIMKEYR_06 <> t$$DIMKEYR_06) OR
        (s$$DIMKEYR_05 <> t$$DIMKEYR_05) OR
        (s$$DIMKEYR_07 <> t$$DIMKEYR_07) OR
        (s$$DIMKEYR_04 <> t$$DIMKEYR_04) OR
        (s$$DIMKEYR_08 <> t$$DIMKEYR_08) OR
```

```
        (s$$DIMKEYR_03 <> t$$DIMKEYR_03) OR
        (s$$DIMKEYR_09 <> t$$DIMKEYR_09) OR
        (s$$DIMKEYR_02 <> t$$DIMKEYR_02) OR
        (s$$DIMKEYR_10 <> t$$DIMKEYR_10) OR
        (s$$DIMKEYR_01 <> t$$DIMKEYR_01)
        )
AND
        (
        (t$$FCTCOL_001 <> 0)
OR      (t$$FCTCOL_002 <> 0)
OR      (t$$FCTCOL_003 <> 0)
OR      (t$$FCTCOL_004 <> 0)
OR      (t$$FCTCOL_005 <> 0)
OR      (t$$FCTCOL_006 <> 0)
OR      (t$$FCTCOL_007 <> 0)
OR      (t$$FCTCOL_008 <> 0)
OR      (t$$FCTCOL_009 <> 0)
OR      (t$$FCTCOL_010 <> 0)
OR      (t$$FCTCOL_011 <> 0)
OR      (t$$FCTCOL_012 <> 0)
OR      (t$$FCTCOL_013 <> 0)
OR      (t$$FCTCOL_014 <> 0)
OR      (t$$FCTCOL_015 <> 0)
OR      (t$$FCTCOL_016 <> 0)
OR      (t$$FCTCOL_017 <> 0)
OR      (t$$FCTCOL_018 <> 0)
OR      (t$$FCTCOL_019 <> 0)
OR      (t$$FCTCOL_020 <> 0)
OR      (t$$FCTCOL_021 <> 0)
OR      (t$$FCTCOL_022 <> 0)
OR      (t$$FCTCOL_023 <> 0)
OR      (t$$FCTCOL_024 <> 0)
        )

--#BLOCK_END# MakeIRM

/*****************************************************************************/
-- Delete the output tables
/*****************************************************************************/

--#BLOCK_BEGIN# DropOutput

$$DDL_BEGIN
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]$$NEXT]
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_INC]
$$DDL_END

--#BLOCK_END# DropOutput

/*****************************************************************************/
--Create FC table in case force_close was
-- not run
/*****************************************************************************/

--#BLOCK_BEGIN# MakeFC

DECLARE $$VAR[fc_exists] $$EPIINT$$EOS

$$DDL_BEGIN_NO_DECLARE

$$VAR_ASSIGN_BEGIN[fc_exists]
SELECT COUNT(1)
$$VAR_ASSIGN_INTO[fc_exists]
FROM $$SQLSERVER[sysobjects]$$ORACLE[tabs]
WHERE
$$SQLSERVER[id = object_id('dbo.$$FCTTBL[]_FC') AND sysstat & 0xf = 3]
$$ORACLE[table_name = UPPER('$$FCTTBL[]_FC')]
$$VAR_ASSIGN_END

$$IF[$$VAR[fc_exists] = 0]
$$DDL_EXEC[
```

```
$$SELECT_INTO_BEGIN[$$FCTTBL[]_FC]
SELECT
        *
$$SELECT_INTO_BODY[$$FCTTBL[]_FC]
FROM
        $$FCTTBL[]$$CURR
WHERE
        1=0
]
$$END_IF
$$DDL_END

--#BLOCK_END# MakeFC

/*******************************************************************************/
-- Create the incremental table
/*******************************************************************************/

--#BLOCK_BEGIN# MakeINC

$$SELECT_INTO_BEGIN[$$FCTTBL[]_INC]
SELECT
        *
$$SELECT_INTO_BODY[$$FCTTBL[]_INC]
FROM $$FCTTBL[]_TIN UNION ALL
SELECT * FROM $$FCTTBL[]_IL UNION ALL
SELECT * FROM $$FCTTBL[]_IR UNION ALL
SELECT * FROM $$FCTTBL[]_IRD UNION ALL
SELECT * FROM $$FCTTBL[]_IND UNION ALL
SELECT * FROM $$FCTTBL[]_IRM UNION ALL
SELECT * FROM $$FCTTBL[]_ILM UNION ALL
SELECT * FROM $$FCTTBL[]_FC UNION ALL
SELECT * FROM $$FCTTBL[]_IDM

--#BLOCK_END# MakeINC

/*******************************************************************************/
-- CR158: We want to load _IMI table and still keep the non-descending
-- order so that the clustered index on a fact table can be created
-- without sorting.  This way can speed up significantly in creating a
-- clustered index on a very large already sorted fact table.
/*******************************************************************************/

--#BLOCK_BEGIN# MakeIMI

$$SELECT_INTO_BEGIN[$$FCTTBL[]_IMI]
SELECT
        *
$$SELECT_INTO_BODY[$$FCTTBL[]_IMI]
FROM $$FCTTBL[]$$CURR
WHERE date_key >= (SELECT MIN(date_key) FROM $$FCTTBL[]_INC)
UNION ALL
SELECT * FROM $$FCTTBL[]_INC
$$SQLSERVER[ORDER BY
        date_key
,       $$DIMKEYR_01
,       $$DIMKEYR_02
,       $$DIMKEYR_03
,       $$DIMKEYR_04
,       $$DIMKEYR_05
,       $$DIMKEYR_06
,       $$DIMKEYR_07
,       $$DIMKEYR_08
,       $$DIMKEYR_09
,       $$DIMKEYR_10
]

--#BLOCK_END# MakeIMI

/*******************************************************************************/
-- Create the new fact table and incremental table
```

```
--
-- Note that transaction tables must be built before
-- these statements are run
/*******************************************************************************/

--#BLOCK_BEGIN# MakeNewFact

$$SELECT_INTO_BEGIN[$$FCTTBL[]$$NEXT]
SELECT *
$$SELECT_INTO_BODY[$$FCTTBL[]$$NEXT]
FROM $$FCTTBL[]$$CURR s
WHERE s.date_key < (SELECT MIN(date_key) FROM $$FCTTBL[]_INC)
UNION ALL
SELECT * FROM $$FCTTBL[]_IMI

--#BLOCK_END# MakeNewFact

/*******************************************************************************/
-- Count processed, inserted rows
/*******************************************************************************/

--#BLOCK_BEGIN# SPResults

DECLARE $$VAR[count_INC] $$EPIINT$$EOS

BEGIN

$$VAR_ASSIGN_BEGIN[count_INC]
SELECT COUNT(1)
$$VAR_ASSIGN_INTO[count_INC]
FROM $$FCTTBL[]_INC
$$VAR_ASSIGN_END

INSERT INTO adaptive_template_profile (token_name, number_rows)
SELECT 'PROCESSED', COUNT(1) FROM $$FCTTBL[]_MFL$$EOS

INSERT INTO adaptive_template_profile (token_name, number_rows)
SELECT 'INSERTED', $$VAR[count_INC] - COUNT(1) FROM $$FCTTBL[]_TIN$$EOS

END$$EOS

--#BLOCK_END# SPResults

/*****************************************************************/
-- Set join order for SQL Server
/*****************************************************************/

--#BLOCK_BEGIN# ForcePlanOff

$$SQLSERVER[SET FORCEPLAN OFF]

--#BLOCK_END# ForcePlanOff

/*******************************************************************************/
-- Drop temp tables and TXN and TIN table
/*******************************************************************************/

--#BLOCK_BEGIN# DropTempsAfter

$$DDL_BEGIN
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_TIN]
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_TMI]
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_FC]
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_TXN]
$$DROP_TABLE_IF_EXISTS[Concat_MFL]
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_1ST]
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_IL]
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_IR]
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_IRD]
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_IND]
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_NFD]
```

```
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_IRM]
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_IDM]
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_ILM]
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_IMI]
$$DDL_END

--#BLOCK_END# DropTempsAfter

--#TEMPLATE_END# load_state
 --#TEMPLATE_BEGIN# load_trans

/*****************************************************************************/
--
-- Copyright * 1997, Epiphany Marketing Software, Inc.  All Rights Reserved.
--
-- load_trans
--
-- Move transaction-like staging data into Fact table - create a temp
-- table with TXN extension that has all old rows along with new rows.
-- Also produce a TIN (TXN INC) table that has only the new rows
--
-- Note that the new table will also include all existing rows from
-- the Fact table.
--
/*****************************************************************************/

/*****************************************************************************/
-- Delete output tables
--
-- Output table is called TXN and includes old and new rows
--
-- Also, leave around _TIN as incremental table from this
-- procedure
--
-- We also create  a table called _TMI which contains all the
-- _TIN records plus the records of overlapping period from the
-- old existing fact table.
/*****************************************************************************/

--#BLOCK_BEGIN# RemoveOutput

$$DDL_BEGIN
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_TXN]
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_TMI]
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_TIN]
$$DDL_END

--#BLOCK_END# RemoveOutput

/*********************************************************************/
-- Set join order for SQL Server
/*********************************************************************/

--#BLOCK_BEGIN# ForcePlanOn

$$SQLSERVER[SET FORCEPLAN ON]

--#BLOCK_END# ForcePlanOn

/*****************************************************************************/
-- Remove stuff already in  fact table
--
-- Note that currently this filter implies that once a transactional
-- fact entry is made it cannot be changed - and no further fact
-- entries on that date or any previous date can be made either
/*****************************************************************************/

--#BLOCK_BEGIN# CreateTIN

$$SELECT_INTO_BEGIN[$$FCTTBL[]_TIN]
SELECT
```

```
        s.iss,
        s.ss_key,
        s.date_key,
        s.transtype_key,
        s.ikey seq
,       s.$$DIMKEYR_01
,       s.$$DIMKEYR_02
,       s.$$DIMKEYR_03
,       s.$$DIMKEYR_04
,       s.$$DIMKEYR_05
,       s.$$DIMKEYR_06
,       s.$$DIMKEYR_07
,       s.$$DIMKEYR_08
,       s.$$DIMKEYR_09
,       s.$$DIMKEYR_10
,       s.$$DEGKEY_01
,       s.$$DEGKEY_02
,       s.$$DEGKEY_03

,       s.$$FCTCOL_001
 ,      s.$$FCTCOL_002
 ,      s.$$FCTCOL_003
 ,      s.$$FCTCOL_004
 ,      s.$$FCTCOL_005
 ,      s.$$FCTCOL_006
 ,      s.$$FCTCOL_007
 ,      s.$$FCTCOL_008
 ,      s.$$FCTCOL_009
 ,      s.$$FCTCOL_010
 ,      s.$$FCTCOL_011
 ,      s.$$FCTCOL_012
 ,      s.$$FCTCOL_013
 ,      s.$$FCTCOL_014
 ,      s.$$FCTCOL_015
 ,      s.$$FCTCOL_016
 ,      s.$$FCTCOL_017
 ,      s.$$FCTCOL_018
 ,      s.$$FCTCOL_019
 ,      s.$$FCTCOL_020
 ,      s.$$FCTCOL_021
 ,      s.$$FCTCOL_022
 ,      s.$$FCTCOL_023
 ,      s.$$FCTCOL_024

$$SELECT_INTO_BODY[$$FCTTBL[]_TIN]
FROM
        $$FSTGTBL[]_MAP s, bus_process b
WHERE
        NOT EXISTS (SELECT * FROM $$FCTTBL[]$$CURR f WHERE
                s.iss = f.iss AND
                s.ss_key = f.ss_key AND
                f.date_key >= s.date_key)
AND     (
        (s.$$FCTCOL_001 <> 0)
 OR     (s.$$FCTCOL_002 <> 0)
 OR     (s.$$FCTCOL_003 <> 0)
 OR     (s.$$FCTCOL_004 <> 0)
 OR     (s.$$FCTCOL_005 <> 0)
 OR     (s.$$FCTCOL_006 <> 0)
 OR     (s.$$FCTCOL_007 <> 0)
 OR     (s.$$FCTCOL_008 <> 0)
 OR     (s.$$FCTCOL_009 <> 0)
 OR     (s.$$FCTCOL_010 <> 0)
 OR     (s.$$FCTCOL_011 <> 0)
 OR     (s.$$FCTCOL_012 <> 0)
 OR     (s.$$FCTCOL_013 <> 0)
 OR     (s.$$FCTCOL_014 <> 0)
 OR     (s.$$FCTCOL_015 <> 0)
 OR     (s.$$FCTCOL_016 <> 0)
 OR     (s.$$FCTCOL_017 <> 0)
 OR     (s.$$FCTCOL_018 <> 0)
```

```
 OR     (s.$$FCTCOL_019 <> 0)
 OR     (s.$$FCTCOL_020 <> 0)
 OR     (s.$$FCTCOL_021 <> 0)
 OR     (s.$$FCTCOL_022 <> 0)
 OR     (s.$$FCTCOL_023 <> 0)
 OR     (s.$$FCTCOL_024 <> 0)
        )
AND
        s.process_key = b.process_key AND b.process_name = 'LoadTrans'

--#BLOCK_END# CreateTIN

/*********************************************************************/
-- Set join order for SQL Server
/*********************************************************************/

--#BLOCK_BEGIN# ForcePlanOff

$$SQLSERVER[SET FORCEPLAN OFF]

--#BLOCK_END# ForcePlanOff

/*****************************************************************************/
-- CR158: We want to load _TMI table and still keep the non-descending
-- order so that the clustered index on a fact table can be created
-- without sorting.  This way can speed up significantly in creating a
-- clustered index on a very large already sorted fact table.
/*****************************************************************************/

--#BLOCK_BEGIN# CreateTMI

$$SELECT_INTO_BEGIN[$$FCTTBL[]_TMI]
SELECT
        *
$$SELECT_INTO_BODY[$$FCTTBL[]_TMI]
FROM
        $$FCTTBL[]$$CURR
WHERE
        date_key >= (SELECT MAX(date_key) FROM $$FCTTBL[]_TIN)
UNION ALL
SELECT
        *
FROM
        $$FCTTBL[]_TIN
$$SQLSERVER[ORDER BY
        date_key
,       $$DIMKEYR_01
,       $$DIMKEYR_02
,       $$DIMKEYR_03
,       $$DIMKEYR_04
,       $$DIMKEYR_05
,       $$DIMKEYR_06
,       $$DIMKEYR_07
,       $$DIMKEYR_08
,       $$DIMKEYR_09
,       $$DIMKEYR_10
]

--#BLOCK_END# CreateTMI

/*****************************************************************************/
-- Insert everything into the new fact table
/*****************************************************************************/

--#BLOCK_BEGIN# CreateTXN

$$SELECT_INTO_BEGIN[$$FCTTBL[]_TXN]
SELECT
        *
$$SELECT_INTO_BODY[$$FCTTBL[]_TXN]
FROM
```

```
       $$FCTTBL[]$$CURR s
WHERE s.date_key < (SELECT MAX(date_key) FROM $$FCTTBL[]_TIN)
UNION ALL
SELECT
       *
FROM
       $$FCTTBL[]_TMI f

--#BLOCK_END# CreateTXN

/*****************************************************************************/
-- Count inserted data and put results into communication table
/*****************************************************************************/

--#BLOCK_BEGIN# SPResults

BEGIN

INSERT INTO adaptive_template_profile (token_name, number_rows)
SELECT 'PROCESSED', COUNT(1) FROM $$FSTGTBL[]_MAP$$EOS

INSERT INTO adaptive_template_profile (token_name, number_rows)
SELECT 'INSERTED', COUNT(1) FROM $$FCTTBL[]_TIN$$EOS

END$$EOS

--#BLOCK_END# SPResults

--#TEMPLATE_END# load_trans

 --#TEMPLATE_BEGIN# index_fact

/*****************************************************************************/
--
-- Copyright * 1997, Epiphany Marketing Software, Inc.  All Rights Reserved.
--
-- Post processing after an extraction run
--
-- Reindex fact tables
-- CR158: added WITH SORTED_DATA in creating cluster index on fact table
--
-- Remove any temp tables generated during the extraction
--
/*****************************************************************************/

/*****************************************************************************/
-- Primary key index the fact table
/*****************************************************************************/

--#BLOCK_BEGIN# PKIndexFact

$$DDL_BEGIN
$$DDL_EXEC[

CREATE UNIQUE INDEX XPK$$FCTTBL[]$$NEXT ON $$FCTTBL[]$$NEXT
 (
  iss , ss_key , date_key , transtype_key , seq
 )
]
$$DDL_END

--#BLOCK_END# PKIndexFact

/*****************************************************************************/
-- Inversion index the fact table
/*****************************************************************************/

--#BLOCK_BEGIN# IEIndexFact

$$DDL_BEGIN
$$DDL_EXEC[
```

```
CREATE $$SQLSERVER[CLUSTERED ]INDEX XIEK$$FCTTBL[]$$NEXT ON $$FCTTBL[]$$NEXT
 (
       date_key
,      $$DIMKEYR_01
,      $$DIMKEYR_02
,      $$DIMKEYR_03
,      $$DIMKEYR_04
,      $$DIMKEYR_05
,      $$DIMKEYR_06
,      $$DIMKEYR_07
,      $$DIMKEYR_08
,      $$DIMKEYR_09
,      $$DIMKEYR_10
 ) $$SQLSERVER[WITH SORTED_DATA]

]

$$DDL_END

--#BLOCK_END# IEIndexFact

/*****************************************************************************/
-- Remove any mapped tables
/*****************************************************************************/

--#BLOCK_BEGIN# RemoveTemps

$$DDL_BEGIN
$$DROP_TABLE_IF_EXISTS[$$FSTGTBL[]_MAP]
$$DDL_END

--#BLOCK_END# RemoveTemps

--#TEMPLATE_END# index_fact
 --#TEMPLATE_BEGIN# ren_trans

/*********************************************************************/
--
-- Copyright * 1997, Epiphany Marketing Software, Inc.  All Rights Reserved.
--
-- ren_trans
--
-- Epiphany Marketing Software, 1997
--
-- Simply change the name of the transaction new table to the
-- actual fact table name - used for Fact tables that don't have
-- any stored procedure other than load_trans attached to them
--
/*********************************************************************/

/*********************************************************************/
-- Delete the output tables
/*********************************************************************/

--#BLOCK_BEGIN# RemoveOutput

$$DDL_BEGIN
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]$$NEXT]
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_INC]
$$DDL_END

--#BLOCK_END# RemoveOutput

/*********************************************************************/
-- Move all transaction rows into the correct new fact table
-- name. Note that we would use sp_rename, except it
-- doesn't work with DB name prefixes
--
-- TBD: Rename instead of re-select
/*********************************************************************/
```

```
--#BLOCK_BEGIN# BuildNewFact

$$SELECT_INTO_BEGIN[$$FCTTBL[]$$NEXT]
SELECT
        *
$$SELECT_INTO_BODY[$$FCTTBL[]$$NEXT]
FROM
        $$FCTTBL[]_TXN

--#BLOCK_END# BuildNewFact

/*************************************************************************/
-- Preserve incremental table
/*************************************************************************/

--#BLOCK_BEGIN# BuildIncremental

$$SELECT_INTO_BEGIN[$$FCTTBL[]_INC]
SELECT
        *
$$SELECT_INTO_BODY[$$FCTTBL[]_INC]
FROM
        $$FCTTBL[]_TIN

--#BLOCK_END# BuildIncremental

/*************************************************************************/
-- Count inserted data and put results into communication table
/*************************************************************************/

--#BLOCK_BEGIN# SPResults

BEGIN

INSERT INTO adaptive_template_profile (token_name, number_rows)
SELECT 'PROCESSED', COUNT(1) FROM $$FCTTBL[]_TXN$$EOS

INSERT INTO adaptive_template_profile (token_name, number_rows)
SELECT 'INSERTED', COUNT(1) FROM $$FCTTBL[]_TXN$$EOS

END$$EOS

--#BLOCK_END# SPResults

/*************************************************************************/
-- Remove temp tables
/*************************************************************************/

--#BLOCK_BEGIN# RemoveTemps

$$DDL_BEGIN
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_TXN]
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_TIN]
$$DROP_TABLE_IF_EXISTS[$$FCTTBL[]_TMI]
$$DDL_END

--#BLOCK_END# RemoveTemps

--#TEMPLATE_END# ren_trans

 --#TEMPLATE_BEGIN# map_keys

/*************************************************************************/
--
-- Copyright * 1997, Epiphany Marketing Software, Inc.  All Rights Reserved.
--
-- map_keys
--
-- Epiphany Marketing Software
--
```

```
-- Map dimension keys from Staging table and report
-- on unjoined rows
--
/*********************************************************************/

/*********************************************************************/
-- Remove output table
/*********************************************************************/

--#BLOCK_BEGIN# DropTemp

$$DDL_BEGIN
$$DROP_TABLE_IF_EXISTS[$$FSTGTBL[]_MAP]
$$DDL_END

--#BLOCK_END# DropTemp

/*********************************************************************/
-- Set join order for SQL Server
/*********************************************************************/

--#BLOCK_BEGIN# ForcePlanOn

$$SQLSERVER[SET FORCEPLAN ON]

--#BLOCK_END# ForcePlanOn

/*********************************************************************/
-- Map dimension keys via Inner joins
/*********************************************************************/

--#BLOCK_BEGIN# MapAll

$$SELECT_INTO_BEGIN[$$FSTGTBL[]_MAP]
SELECT
        s.iss,
        s.ss_key,
        s.date_key,
        s.transtype_key,
        s.ikey,
        s.process_key
,       $$PIPE_STATE

,       m_04.$$DIMKEY_04 $$DIMKEYR_04
,       m_03.$$DIMKEY_03 $$DIMKEYR_03
,       m_06.$$DIMKEY_06 $$DIMKEYR_06
,       m_02.$$DIMKEY_02 $$DIMKEYR_02
,       m_08.$$DIMKEY_08 $$DIMKEYR_08
,       m_05.$$DIMKEY_05 $$DIMKEYR_05
,       m_09.$$DIMKEY_09 $$DIMKEYR_09
,       m_01.$$DIMKEY_01 $$DIMKEYR_01
,       m_07.$$DIMKEY_07 $$DIMKEYR_07
,       m_10.$$DIMKEY_10 $$DIMKEYR_10

,       $$DEGKEY_03
,       $$DEGKEY_02
,       $$DEGKEY_01

,       s.$$FCTCOL_001
,       s.$$FCTCOL_002
,       s.$$FCTCOL_003
,       s.$$FCTCOL_004
,       s.$$FCTCOL_005
,       s.$$FCTCOL_006
,       s.$$FCTCOL_007
,       s.$$FCTCOL_008
,       s.$$FCTCOL_009
,       s.$$FCTCOL_010
,       s.$$FCTCOL_011
,       s.$$FCTCOL_012
,       s.$$FCTCOL_013
```

```
,       s.$$FCTCOL_014
,       s.$$FCTCOL_015
,       s.$$FCTCOL_016
,       s.$$FCTCOL_017
,       s.$$FCTCOL_018
,       s.$$FCTCOL_019
,       s.$$FCTCOL_020
,       s.$$FCTCOL_021
,       s.$$FCTCOL_022
,       s.$$FCTCOL_023
,       s.$$FCTCOL_024

$$SELECT_INTO_BODY[$$FSTGTBL[]_MAP]
FROM
        $$FSTGTBL[] s
,       $$MAPTBL_04$$NEXT m_04 $$SQLSERVER[(index = 1)]
,       $$MAPTBL_03$$NEXT m_03 $$SQLSERVER[(index = 1)]
,       $$MAPTBL_06$$NEXT m_06 $$SQLSERVER[(index = 1)]
,       $$MAPTBL_02$$NEXT m_02 $$SQLSERVER[(index = 1)]
,       $$MAPTBL_08$$NEXT m_08 $$SQLSERVER[(index = 1)]
,       $$MAPTBL_05$$NEXT m_05 $$SQLSERVER[(index = 1)]
,       $$MAPTBL_09$$NEXT m_09 $$SQLSERVER[(index = 1)]
,       $$MAPTBL_01$$NEXT m_01 $$SQLSERVER[(index = 1)]
,       $$MAPTBL_07$$NEXT m_07 $$SQLSERVER[(index = 1)]
,       $$MAPTBL_10$$NEXT m_10 $$SQLSERVER[(index = 1)]

WHERE 1=1
AND     m_04.iss = s.iss AND m_04.$$DSTGKEY_04 = s.$$DSTGKEYR_04
AND     m_03.iss = s.iss AND m_03.$$DSTGKEY_03 = s.$$DSTGKEYR_03
AND     m_06.iss = s.iss AND m_06.$$DSTGKEY_06 = s.$$DSTGKEYR_06
AND     m_02.iss = s.iss AND m_02.$$DSTGKEY_02 = s.$$DSTGKEYR_02
AND     m_08.iss = s.iss AND m_08.$$DSTGKEY_08 = s.$$DSTGKEYR_08
AND     m_05.iss = s.iss AND m_05.$$DSTGKEY_05 = s.$$DSTGKEYR_05
AND     m_09.iss = s.iss AND m_09.$$DSTGKEY_09 = s.$$DSTGKEYR_09
AND     m_01.iss = s.iss AND m_01.$$DSTGKEY_01 = s.$$DSTGKEYR_01
AND     m_07.iss = s.iss AND m_07.$$DSTGKEY_07 = s.$$DSTGKEYR_07
AND     m_10.iss = s.iss AND m_10.$$DSTGKEY_10 = s.$$DSTGKEYR_10

--#BLOCK_END# MapAll

/*****************************************************************/
-- Set join order for SQL Server
/*****************************************************************/

--#BLOCK_BEGIN# ForcePlanOff

$$SQLSERVER[SET FORCEPLAN OFF]

--#BLOCK_END# ForcePlanOff

/*****************************************************************/
-- Look for unjoined data, Report on processed rows
/*****************************************************************/

--#BLOCK_BEGIN# SPResults

$$DECLARE_BEGIN
$$DECLARE_BODY[$$VAR[unjoined] $$EPIINT]
$$DECLARE_BODY[$$VAR[processed] $$EPIINT]

BEGIN

$$VAR_ASSIGN_BEGIN[processed]
SELECT COUNT(1)
$$VAR_ASSIGN_INTO[processed]
FROM $$FSTGTBL[]
$$VAR_ASSIGN_END

$$VAR_ASSIGN_BEGIN[unjoined]
SELECT $$VAR[processed] - COUNT(1)
$$VAR_ASSIGN_INTO[unjoined]
```

```
FROM $$FSTGTBL[]_MAP
$$VAR_ASSIGN_END

INSERT INTO adaptive_template_profile (token_name, number_rows)
SELECT 'UNJOINED', $$VAR[unjoined] $$NO_FROM_LIST$$EOS

INSERT INTO adaptive_template_profile (token_name, number_rows)
SELECT 'PROCESSED', $$VAR[processed] $$NO_FROM_LIST$$EOS

INSERT INTO adaptive_template_profile (token_name, number_rows)
SELECT 'INSERTED', $$VAR[processed] - $$VAR[unjoined]  $$NO_FROM_LIST$$EOS

END$$EOS

--#BLOCK_END# SPResults

/*********************************************************************/
-- Index this temp table
/*********************************************************************/

--#BLOCK_BEGIN# IndexMap

$$DDL_BEGIN
$$DDL_EXEC[
CREATE INDEX X$$FSTGTBL[]_MAP ON $$FSTGTBL[]_MAP
(
 iss, ss_key, date_key, ikey
)
]
$$DDL_END

--#BLOCK_END# IndexMap

--#TEMPLATE_END# map_keys
 --#TEMPLATE_BEGIN# upd_unj

/*********************************************************************/
--
-- Copyright * 1997, Epiphany Marketing Software, Inc.  All Rights Reserved
--
-- upd_unj
--
-- Epiphany Marketing Software
--
-- Update all dimension keys to 'UNKNOWN' in staging table
-- where referential integrity fails
--
/*********************************************************************/

/*********************************************************************/
-- Count the number of rows to update in the staging table - that is, those
-- that have at least one Foreign key where referential integrity fails
/*********************************************************************/

--#BLOCK_BEGIN# CountUnj

BEGIN

INSERT INTO adaptive_template_profile (token_name, number_rows)
SELECT 'PROCESSED', COUNT(1) FROM $$FSTGTBL[]$$EOS

INSERT INTO adaptive_template_profile (token_name, number_rows)
SELECT 'MODIFIED', COUNT(1)
FROM
        $$FSTGTBL[] s
WHERE 1=0
OR NOT EXISTS (SELECT 1 FROM $$MAPTBL_04$$NEXT m_04 WHERE m_04.iss = s.iss AND
m_04.$$DSTGKEY_04 = $$DSTGKEYR_04)
OR NOT EXISTS (SELECT 1 FROM $$MAPTBL_03$$NEXT m_03 WHERE m_03.iss = s.iss AND
m_03.$$DSTGKEY_03 = $$DSTGKEYR_03)
OR NOT EXISTS (SELECT 1 FROM $$MAPTBL_06$$NEXT m_06 WHERE m_06.iss = s.iss AND
```

```
m_06.$$DSTGKEY_06 = $$DSTGKEYR_06)
OR NOT EXISTS (SELECT 1 FROM $$MAPTBL_02$$NEXT m_02 WHERE m_02.iss = s.iss AND
m_02.$$DSTGKEY_02 = $$DSTGKEYR_02)
OR NOT EXISTS (SELECT 1 FROM $$MAPTBL_08$$NEXT m_08 WHERE m_08.iss = s.iss AND
m_08.$$DSTGKEY_08 = $$DSTGKEYR_08)
OR NOT EXISTS (SELECT 1 FROM $$MAPTBL_05$$NEXT m_05 WHERE m_05.iss = s.iss AND
m_05.$$DSTGKEY_05 = $$DSTGKEYR_05)
OR NOT EXISTS (SELECT 1 FROM $$MAPTBL_09$$NEXT m_09 WHERE m_09.iss = s.iss AND
m_09.$$DSTGKEY_09 = $$DSTGKEYR_09)
OR NOT EXISTS (SELECT 1 FROM $$MAPTBL_01$$NEXT m_01 WHERE m_01.iss = s.iss AND
m_01.$$DSTGKEY_01 = $$DSTGKEYR_01)
OR NOT EXISTS (SELECT 1 FROM $$MAPTBL_07$$NEXT m_07 WHERE m_07.iss = s.iss AND
m_07.$$DSTGKEY_07 = $$DSTGKEYR_07)
OR NOT EXISTS (SELECT 1 FROM $$MAPTBL_10$$NEXT m_10 WHERE m_10.iss = s.iss AND
m_10.$$DSTGKEY_10 = $$DSTGKEYR_10)
$$EOS
END$$EOS

--#BLOCK_END# CountUnj

/*********************************************************************/
-- Update foreign keys where referential integrity fails
/*********************************************************************/

--#BLOCK_BEGIN# UpdateUnj$$DSTGKEYR_04

UPDATE $$FSTGTBL[] SET $$DSTGKEYR_04 = 'UNKNOWN'
WHERE NOT EXISTS (SELECT 1 FROM $$MAPTBL_04$$NEXT m
WHERE m.iss = $$FSTGTBL[].iss AND m.$$DSTGKEY_04 = $$FSTGTBL[].$$DSTGKEYR_04)

--#BLOCK_END# UpdateUnj$$DSTGKEYR_04

--#BLOCK_BEGIN# UpdateUnj$$DSTGKEYR_03

UPDATE $$FSTGTBL[] SET $$DSTGKEYR_03 = 'UNKNOWN'
WHERE NOT EXISTS (SELECT 1 FROM $$MAPTBL_03$$NEXT m
WHERE m.iss = $$FSTGTBL[].iss AND m.$$DSTGKEY_03 = $$FSTGTBL[].$$DSTGKEYR_03)

--#BLOCK_END# UpdateUnj$$DSTGKEYR_03

--#BLOCK_BEGIN# UpdateUnj$$DSTGKEYR_06

UPDATE $$FSTGTBL[] SET $$DSTGKEYR_06 = 'UNKNOWN'
WHERE NOT EXISTS (SELECT 1 FROM $$MAPTBL_06$$NEXT m
WHERE m.iss = $$FSTGTBL[].iss AND m.$$DSTGKEY_06 = $$FSTGTBL[].$$DSTGKEYR_06)

--#BLOCK_END# UpdateUnj$$DSTGKEYR_06

--#BLOCK_BEGIN# UpdateUnj$$DSTGKEYR_02

UPDATE $$FSTGTBL[] SET $$DSTGKEYR_02 = 'UNKNOWN'
WHERE NOT EXISTS (SELECT 1 FROM $$MAPTBL_02$$NEXT m
WHERE m.iss = $$FSTGTBL[].iss AND m.$$DSTGKEY_02 = $$FSTGTBL[].$$DSTGKEYR_02)

--#BLOCK_END# UpdateUnj$$DSTGKEYR_02

--#BLOCK_BEGIN# UpdateUnj$$DSTGKEYR_08

UPDATE $$FSTGTBL[] SET $$DSTGKEYR_08 = 'UNKNOWN'
WHERE NOT EXISTS (SELECT 1 FROM $$MAPTBL_08$$NEXT m
WHERE m.iss = $$FSTGTBL[].iss AND m.$$DSTGKEY_08 = $$FSTGTBL[].$$DSTGKEYR_08)

--#BLOCK_END# UpdateUnj$$DSTGKEYR_08

--#BLOCK_BEGIN# UpdateUnj$$DSTGKEYR_05

UPDATE $$FSTGTBL[] SET $$DSTGKEYR_05 = 'UNKNOWN'
WHERE NOT EXISTS (SELECT 1 FROM $$MAPTBL_05$$NEXT m
WHERE m.iss = $$FSTGTBL[].iss AND m.$$DSTGKEY_05 = $$FSTGTBL[].$$DSTGKEYR_05)

--#BLOCK_END# UpdateUnj$$DSTGKEYR_05
```

```
--#BLOCK_BEGIN# UpdateUnj$$DSTGKEYR_09

UPDATE $$FSTGTBL[] SET $$DSTGKEYR_09 = 'UNKNOWN'
WHERE NOT EXISTS (SELECT 1 FROM $$MAPTBL_09$$NEXT m
WHERE m.iss = $$FSTGTBL[].iss AND m.$$DSTGKEY_09 = $$FSTGTBL[].$$DSTGKEYR_09)

--#BLOCK_END# UpdateUnj$$DSTGKEYR_09

--#BLOCK_BEGIN# UpdateUnj$$DSTGKEYR_01

UPDATE $$FSTGTBL[] SET $$DSTGKEYR_01 = 'UNKNOWN'
WHERE NOT EXISTS (SELECT 1 FROM $$MAPTBL_01$$NEXT m
WHERE m.iss = $$FSTGTBL[].iss AND m.$$DSTGKEY_01 = $$FSTGTBL[].$$DSTGKEYR_01)

--#BLOCK_END# UpdateUnj$$DSTGKEYR_01

--#BLOCK_BEGIN# UpdateUnj$$DSTGKEYR_07

UPDATE $$FSTGTBL[] SET $$DSTGKEYR_07 = 'UNKNOWN'
WHERE NOT EXISTS (SELECT 1 FROM $$MAPTBL_07$$NEXT m
WHERE m.iss = $$FSTGTBL[].iss AND m.$$DSTGKEY_07 = $$FSTGTBL[].$$DSTGKEYR_07)

--#BLOCK_END# UpdateUnj$$DSTGKEYR_07

--#BLOCK_BEGIN# UpdateUnj$$DSTGKEYR_10

UPDATE $$FSTGTBL[] SET $$DSTGKEYR_10 = 'UNKNOWN'
WHERE NOT EXISTS (SELECT 1 FROM $$MAPTBL_10$$NEXT m
WHERE m.iss = $$FSTGTBL[].iss AND m.$$DSTGKEY_10 = $$FSTGTBL[].$$DSTGKEYR_10)

--#BLOCK_END# UpdateUnj$$DSTGKEYR_10


--#TEMPLATE_END# upd_unj
```

The following are the post-parsed SQL source for the adaptive templates as filled in with corresponding schema definitions.

```
 --#TEMPLATE_BEGIN# force_close

/*****************************************************************************/
--
-- Copyright * 1997, Epiphany Marketing Software, Inc.  All Rights Reserved.
--
-- force_close
--
-- Close out deleted orders - those that no longer appear in the
-- staging table
--
-- SEE SAFETY VALVE BELOW
--
/*****************************************************************************/

/*****************************************************************************/
-- Delete temporary tables
```

```
/*****************************************************************************/

--#BLOCK_BEGIN# DropTemps

IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_FC') AND sysstat & 0xf =
3) DROP TABLE Order_0_FC

--#BLOCK_END# DropTemps

/*****************************************************************************/
-- Insert negative BOOKs for deleted orders
--
-- FC: ForceClose
/*****************************************************************************/

--#BLOCK_BEGIN# MakeFC

SELECT
        f.iss,
        f.ss_key,
        MAX(f.date_key) date_key,
        MIN(f.transtype_key) transtype_key,
        MAX(f.seq) + 1 seq
,       f.customerbillto_key
,       f.product_key
,       f.application_key
,       f.program_key
,       f.customershipto_key
,       f.territory_key
,       f.warehouse_key

,       -SUM(f.net_price) net_price
,       -SUM(f.number_units) number_units

INTO Order_0_FC
FROM
        Order_0_A f
WHERE
        NOT EXISTS
        (SELECT 1 FROM OrderStage_MAP s WHERE s.iss = f.iss AND s.ss_key = f.ss_key)
GROUP BY
        f.iss,
        f.ss_key
,       f.customerbillto_key
,       f.product_key
,       f.application_key
,       f.program_key
,       f.customershipto_key
,       f.territory_key
,       f.warehouse_key

HAVING
        (
        (SUM(f.net_price) <> 0)
 OR     (SUM(f.number_units) <> 0)
        )
AND
        MIN(f.transtype_key) <= 99
AND
        MIN(f.transtype_key) >= 1

--#BLOCK_END# MakeFC

/*****************************************************************************/
-- SAFETY VALVE - THIS PROC ONLY DOES ANYTHING
-- IF THE STAGING TABLE HAS AT LEAST ONE ROW
/*****************************************************************************/
```

```
--#BLOCK_BEGIN# SafetyValue

DECLARE @count_MAP INT

BEGIN

SELECT @count_MAP = (
SELECT COUNT(1)

FROM OrderStage_MAP
)

IF ((@count_MAP = 0))
DELETE FROM Order_0_FC


END

--#BLOCK_END# SafetyValue

/*****************************************************************************/
-- Count processed, inserted rows
/*****************************************************************************/

--#BLOCK_BEGIN# SPResults

BEGIN

INSERT INTO adaptive_template_profile (token_name, number_rows)
SELECT 'PROCESSED', COUNT(1) FROM Order_0_A

INSERT INTO adaptive_template_profile (token_name, number_rows)
SELECT 'INSERTED', COUNT(1) FROM Order_0_FC

END

--#BLOCK_END# SPResults

--#TEMPLATE_END# force_close


 --#TEMPLATE_BEGIN# load_state

/*****************************************************************************/
--
-- Copyright * 1997, Epiphany Marketing Software, Inc.  All Rights Reserved.
--
-- load_state
--
-- Load order bookings into fact table by creating transactional
-- data from state data
--
-- load_trans must be run before this procedure to create TIN table
--
/*****************************************************************************/

/*****************************************************************************/
-- Delete temporary tables
/*****************************************************************************/

--#BLOCK_BEGIN# DropTemps


IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_MFL') AND sysstat & 0xf
= 3) DROP TABLE Order_0_MFL
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_1ST') AND sysstat & 0xf
= 3) DROP TABLE Order_0_1ST
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_IL') AND sysstat & 0xf =
3) DROP TABLE Order_0_IL
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_IR') AND sysstat & 0xf =
3) DROP TABLE Order_0_IR
```

```
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_IRD') AND sysstat & 0xf
= 3) DROP TABLE Order_0_IRD
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_IND') AND sysstat & 0xf
= 3) DROP TABLE Order_0_IND
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_NFD') AND sysstat & 0xf
= 3) DROP TABLE Order_0_NFD
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_IRM') AND sysstat & 0xf
= 3) DROP TABLE Order_0_IRM
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_IDM') AND sysstat & 0xf
= 3) DROP TABLE Order_0_IDM
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_ILM') AND sysstat & 0xf
= 3) DROP TABLE Order_0_ILM
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_IMI') AND sysstat & 0xf
= 3) DROP TABLE Order_0_IMI


--#BLOCK_END# DropTemps

/*****************************************************************/
-- Set join order for SQL Server
/*****************************************************************/

--#BLOCK_BEGIN# ForcePlanOn

SET FORCEPLAN ON

--#BLOCK_END# ForcePlanOn

/*****************************************************************************************/
-- Remove rows older than fact table - history can not be rewritten - only
-- the last date for an order can be changed. Note that we compare transtype's
-- because SHIP type transactions might occur at a later date and we don't want
-- those to interfere
--
-- Also, since the staging table may have multiple entries for a given order on
-- a single day - we assume that the list one inserted in the Staging table will
-- be used (since ikey is an IDENTITY column)
--
-- Note that a given ss_key must use the same Booking transtype for all of time,
-- otherwise the transtype_key
--
-- MFL: Mapped Filtered
/*****************************************************************************************/

--#BLOCK_BEGIN# MakeMFL


SELECT
        s.*
INTO Order_0_MFL
FROM
        OrderStage_MAP s, bus_process b
WHERE
        ((s.date_key >= (SELECT MAX(date_key) FROM Order_0_A f WHERE
                s.iss = f.iss AND s.ss_key = f.ss_key AND
                s.transtype_key = f.transtype_key))
        OR NOT EXISTS (SELECT * FROM Order_0_A f WHERE
                s.iss = f.iss AND s.ss_key = f.ss_key AND
                s.transtype_key = f.transtype_key))
AND     s.ikey = (SELECT MAX(t.ikey) FROM OrderStage_MAP t WHERE
                s.iss = t.iss AND
                s.ss_key = t.ss_key AND
                s.date_key = t.date_key AND
                t.process_key = b.process_key)
AND
        s.process_key = b.process_key AND b.process_name = 'LoadState'

--#BLOCK_END# MakeMFL

/*****************************************************************************************/
-- Index MFL table for later queries
```

```
/*****************************************************************************/

--#BLOCK_BEGIN# IndexMFL


EXEC('
CREATE INDEX XOrder_0_MFL ON Order_0_MFL
(
 iss, ss_key, date_key
)
')


--#BLOCK_END# IndexMFL

/*****************************************************************************/
-- Get oldest state rows for each unique sskey
--
-- We need to treat the first entry for each order
-- in the staging table separately from all others, since
-- only the first entry needs to be compared with
-- already existing fact entry rows to create transactions.
-- All subsequent dates for that order in the Fact table
-- can be delta'd with other staging table entries - see the
-- section below on Pairwise deltas.
--
-- MFL should be indexed
--
-- 1ST: The first record for each iss, ss_key
/*****************************************************************************/

--#BLOCK_BEGIN# Make1ST


SELECT
        s.*
INTO Order_0_1ST
FROM
        Order_0_MFL s
WHERE
        s.date_key = (SELECT MIN(date_key) FROM Order_0_MFL t WHERE
                s.iss = t.iss AND s.ss_key = t.ss_key)

--#BLOCK_END# Make1ST

/*****************************************************************************/
-- Index 1ST for later queries
/*****************************************************************************/

--#BLOCK_BEGIN# Index1ST


EXEC('
CREATE UNIQUE INDEX XPKOrder_0_1ST ON Order_0_1ST
(
 iss, ss_key
)
')


--#BLOCK_END# Index1ST

/*****************************************************************************/
-- Insert negative BOOKs for changed dim keys
--
-- This query will add up all existing Books and Loss's
-- for this order and the net facts will be cancelled out
-- with the old Dimension keys. Note that an invariant of this
-- procedure is that only one set of dimensions at a time
-- can have non-zero facts.
--
```

```
-- Fact table Should be indexed
--
-- HAVING Clause is needed to prevent changing of dimensions
-- on fully shipped order from causing a transaction - no sense
-- creating fact rows with all zero's in them
--
-- Note that we increment the sequence number just in case
-- this new transaction occurs on the same date as the last
-- existing one in the fact table - to avoid index errors
--
-- IL: InsertLost
/*******************************************************************************/

--#BLOCK_BEGIN# MakeIL


SELECT
        s.iss,
        s.ss_key,
        s.date_key,
        s.transtype_key,
        MAX(f.seq) + 1 seq
,       f.customerbillto_key
,       f.product_key
,       f.application_key
,       f.program_key
,       f.customershipto_key
,       f.territory_key
,       f.warehouse_key

,       -SUM(f.net_price) net_price
,       -SUM(f.number_units) number_units

INTO Order_0_IL
FROM
        Order_0_1ST s, Order_0_A f
WHERE
        s.iss = f.iss AND s.ss_key = f.ss_key
AND
        ((s.territory_key <> f.territory_key) OR
        (s.customershipto_key <> f.customershipto_key) OR
        (s.warehouse_key <> f.warehouse_key) OR
        (s.program_key <> f.program_key) OR
        (s.application_key <> f.application_key) OR
        (s.product_key <> f.product_key) OR
        (s.customerbillto_key <> f.customerbillto_key) )
GROUP BY
        s.iss,
        s.ss_key,
        s.date_key,
        s.transtype_key
,       f.customerbillto_key
,       f.product_key
,       f.application_key
,       f.program_key
,       f.customershipto_key
,       f.territory_key
,       f.warehouse_key

HAVING
        MIN(f.transtype_key) = s.transtype_key
AND
        (
        (SUM(f.net_price) <> 0)
OR      (SUM(f.number_units) <> 0)
        )

--#BLOCK_END# MakeIL

/*******************************************************************************/
-- Index IL for later queries
```

```
/*****************************************************************************/

--#BLOCK_BEGIN# IndexIL


EXEC('
CREATE INDEX XPKOrder_0_IL ON Order_0_IL
(
 iss, ss_key
)
')


--#BLOCK_END# IndexIL

/*****************************************************************************/
-- Insert BOOKs for changed dim keys
--
-- When a dimension changes then just create a booking
-- transaction for whatever we negated above with the new
-- dimension and fact values
--
-- 1ST shoud be indexed
--
-- Note that we add one to whatever we used as the last
-- seq because this transaction occurs on the same
-- date as the negative one above
--
-- IR: Insert Rebook
/*****************************************************************************/

--#BLOCK_BEGIN# MakeIR


SELECT
        s.iss,
        s.ss_key,
        s.date_key,
        l.transtype_key,
        l.seq + 1 seq
,       s.customerbillto_key
,       s.product_key
,       s.application_key
,       s.program_key
,       s.customershipto_key
,       s.territory_key
,       s.warehouse_key

,       -l.net_price net_price
,       -l.number_units number_units

INTO Order_0_IR
FROM
        Order_0_IL l, Order_0_1ST s
WHERE l.iss = s.iss AND l.ss_key = s.ss_key

--#BLOCK_END# MakeIR

/*****************************************************************************/
-- Insert BOOKs for changed dim keys where fact
-- also changed
--
-- When a dimension changes at the same time as
-- a fact then we need to make up the fact difference
--
-- 1ST shoud be indexed
--
-- Note that we add two to whatever we used as the last
-- seq because this transaction occurs on the same
-- date as the negative and positive ones above
--
```

```
-- Note also that the Left Outer join uses transtype_key
-- so that only the Bookings at the old value will be counted.
-- Whereas above for the negative transaction value
-- we want to include Shipments in our calculation, here
-- we only want to see how Booking Facts have changed.
--
-- Here again, only one Booking transaction type is supported
-- per ss_key
--
-- IRD: Insert Rebook delta
/*****************************************************************************/

--#BLOCK_BEGIN# MakeIRD


SELECT
       s.iss,
       s.ss_key,
       s.date_key,
       s.transtype_key,
       l.seq + 2 seq
,      s.customerbillto_key
,      s.product_key
,      s.application_key
,      s.program_key
,      s.customershipto_key
,      s.territory_key
,      s.warehouse_key

,      MAX(s.net_price)-ISNULL(SUM(f.net_price) , 0) net_price
,      MAX(s.number_units)-ISNULL(SUM(f.number_units) , 0) number_units

INTO Order_0_IRD
FROM
       Order_0_IL l, Order_0_1ST s
       LEFT OUTER JOIN Order_0_A f  ON  s.iss = f.iss AND s.ss_key = f.ss_key AND
s.transtype_key = f.transtype_key
WHERE
       l.iss = s.iss AND l.ss_key = s.ss_key

GROUP BY
       s.iss,
       s.ss_key,
       s.date_key,
       s.transtype_key,
       l.seq
,      s.customerbillto_key
,      s.product_key
,      s.application_key
,      s.program_key
,      s.customershipto_key
,      s.territory_key
,      s.warehouse_key

HAVING
       (ISNULL(SUM(f.net_price) , 0) <> MAX(s.net_price))
OR     (ISNULL(SUM(f.number_units) , 0) <> MAX(s.number_units))

--#BLOCK_END# MakeIRD

/*****************************************************************************/
-- Insert BOOKs for deltas with same dim keys OR for
-- brand new orders.
--
-- Note that we DON'T want to count Shipments
-- (so shipment ss_key's should be different from
-- order ss_keys) since we just want bookings to sum up
-- to whatever this transcation says they should be.
--
-- Fact table should be indexed
--
```

```
-- WHERE clause prevents double booking on changed
-- dimension - if we didn't use the NOT EXISTS clause
-- then this query would repeat the work of the last one
-- above - which we have already taken care of
--
-- HAVING clause ensures that multiple 0 records don't
-- get inserted whenever this procedure is run
--
-- Note that we increment the sequence number just in case
-- this new transaction occurs on the same date as the last
-- existing one in the fact table - to avoid index errors
--
-- IND: Insert New Delta
/*****************************************************************************/

--#BLOCK_BEGIN# MakeIND


SELECT
        s.iss,
        s.ss_key,
        s.date_key,
        s.transtype_key,
        ISNULL(MAX(f.seq) , 0) + 1 seq
,       s.customerbillto_key
,       s.product_key
,       s.application_key
,       s.program_key
,       s.customershipto_key
,       s.territory_key
,       s.warehouse_key

,       MAX(s.net_price)-ISNULL(SUM(f.net_price) , 0) net_price
,       MAX(s.number_units)-ISNULL(SUM(f.number_units) , 0) number_units

INTO Order_0_IND
FROM
        Order_0_1ST s LEFT OUTER JOIN Order_0_A f  ON
                s.iss = f.iss AND s.ss_key = f.ss_key AND f.transtype_key = s.transtype_key
WHERE
        NOT EXISTS (SELECT * FROM Order_0_IL WHERE iss = s.iss AND ss_key = s.ss_key)

GROUP BY
        s.iss,
        s.ss_key,
        s.date_key,
        s.transtype_key
,       s.customerbillto_key
,       s.product_key
,       s.application_key
,       s.program_key
,       s.customershipto_key
,       s.territory_key
,       s.warehouse_key

HAVING
        (ISNULL(SUM(f.net_price) , 0) <> MAX(s.net_price))
 OR     (ISNULL(SUM(f.number_units) , 0) <> MAX(s.number_units))


--#BLOCK_END# MakeIND

/*****************************************************************************/
-- Form pairwise deltas for all rows except earliest for each sskey
--
-- Each row created in NFD will consist of two sequential entries from the
-- staing table. So if N enties for an order exist in MFL (after we have filtered
-- out same-date duplicates) then all the queries above will deal with the earliest entry,
whereas
-- all the queries below (including this one) will deal with the N-1 deltaing transactions
--
```

```
-- This query assumes that MFL will already have been filtered
-- to have a single record for each order/datekey
--
-- NFD: Not First Delta
/*****************************************************************************/

--#BLOCK_BEGIN# MakeNFD


SELECT
        s.iss siss, t.iss tiss
,       s.ss_key sss_key, t.ss_key tss_key
,       s.date_key sdate_key, t.date_key tdate_key
,       s.transtype_key stranstype_key, t.transtype_key ttranstype_key
,       s.customerbillto_key scustomerbillto_key, t.customerbillto_key tcustomerbillto_key
,       s.product_key sproduct_key, t.product_key tproduct_key
,       s.application_key sapplication_key, t.application_key tapplication_key
,       s.program_key sprogram_key, t.program_key tprogram_key
,       s.customershipto_key scustomershipto_key, t.customershipto_key tcustomershipto_key
,       s.territory_key sterritory_key, t.territory_key tterritory_key
,       s.warehouse_key swarehouse_key, t.warehouse_key twarehouse_key
,       s.net_price snet_price, t.net_price tnet_price
,       s.number_units snumber_units, t.number_units tnumber_units

INTO Order_0_NFD
FROM
        Order_0_MFL s, Order_0_MFL t
WHERE
        s.iss = t.iss AND s.ss_key = t.ss_key
AND
        s.date_key = (SELECT MAX(date_key) FROM Order_0_MFL u WHERE
        u.iss = s.iss AND u.ss_key = s.ss_key AND u.date_key < t.date_key)

--#BLOCK_END# MakeNFD

/*****************************************************************************/
--
-- Insert BOOKs for deltas with same dim keys
--
-- If the dimensions don't change then we create a
-- new booking order (as long as at least one of the facts
-- have changed)
--
-- IDM: Insert Delta More
--
/*****************************************************************************/

--#BLOCK_BEGIN# MakeIDM


SELECT
        tiss iss,
        tss_key ss_key,
        tdate_key date_key,
        ttranstype_key transtype_key,
        0 seq
,       tcustomerbillto_key customerbillto_key
,       tproduct_key product_key
,       tapplication_key application_key
,       tprogram_key program_key
,       tcustomershipto_key customershipto_key
,       tterritory_key territory_key
,       twarehouse_key warehouse_key

,       tnet_price-snet_price net_price
 ,      tnumber_units-snumber_units number_units

INTO Order_0_IDM
FROM
        Order_0_NFD d
WHERE
```

```
        (
        (sterritory_key = tterritory_key) AND
        (scustomershipto_key = tcustomershipto_key) AND
        (swarehouse_key = twarehouse_key) AND
        (sprogram_key = tprogram_key) AND
        (sapplication_key = tapplication_key) AND
        (sproduct_key = tproduct_key) AND
        (scustomerbillto_key = tcustomerbillto_key)
        )
AND
        (
        (snet_price <> tnet_price)
OR      (snumber_units <> tnumber_units)
        )

--#BLOCK_END# MakeIDM

/*****************************************************************************/
--
-- Insert negative BOOKs for deltas with different dim keys
--
-- If one of the dimensions change then we first create a lose transaction for
-- all the previous facts. (Negate all the facts from the earlier of the two
-- transactions)
--
-- ILM: Insert Lost More
--
/*****************************************************************************/

--#BLOCK_BEGIN# MakeILM


SELECT
        siss iss,
        sss_key ss_key,
        tdate_key date_key,
        stranstype_key transtype_key,
        0 seq
,       scustomerbillto_key customerbillto_key
,       sproduct_key product_key
,       sapplication_key application_key
,       sprogram_key program_key
,       scustomershipto_key customershipto_key
,       sterritory_key territory_key
,       swarehouse_key warehouse_key

,       -snet_price net_price
 ,      -snumber_units number_units

INTO Order_0_ILM
FROM
        Order_0_NFD d
WHERE
        (
        (sterritory_key <> tterritory_key) OR
        (scustomershipto_key <> tcustomershipto_key) OR
        (swarehouse_key <> twarehouse_key) OR
        (sprogram_key <> tprogram_key) OR
        (sapplication_key <> tapplication_key) OR
        (sproduct_key <> tproduct_key) OR
        (scustomerbillto_key <> tcustomerbillto_key)
        )
AND
        (
        (snet_price <> 0)
OR      (snumber_units <> 0)
        )

--#BLOCK_END# MakeILM

/*****************************************************************************/
```

```
--
-- Insert BOOKs for deltas with different dim keys
--
-- When a dimension key changes then we can simply insert all the new facts with the
-- new dimension keys
--
-- Note that seq = 1 here because this is the second transaction on this date for
-- this order.
--
-- IRM: Insert Rebook More
--
/*****************************************************************************/

--#BLOCK_BEGIN# MakeIRM


SELECT
        tiss iss,
        tss_key ss_key,
        tdate_key date_key,
        ttranstype_key transtype_key,
        1 seq
,       tcustomerbillto_key customerbillto_key
,       tproduct_key product_key
,       tapplication_key application_key
,       tprogram_key program_key
,       tcustomershipto_key customershipto_key
,       tterritory_key territory_key
,       twarehouse_key warehouse_key

,       tnet_price net_price
 ,      tnumber_units number_units

INTO Order_0_IRM
FROM
        Order_0_NFD d
WHERE
        (
        (sterritory_key <> tterritory_key) OR
        (scustomershipto_key <> tcustomershipto_key) OR
        (swarehouse_key <> twarehouse_key) OR
        (sprogram_key <> tprogram_key) OR
        (sapplication_key <> tapplication_key) OR
        (sproduct_key <> tproduct_key) OR
        (scustomerbillto_key <> tcustomerbillto_key)
        )
AND
        (
        (tnet_price <> 0)
OR      (tnumber_units <> 0)
        )

--#BLOCK_END# MakeIRM

/*****************************************************************************/
-- Delete the output tables
/*****************************************************************************/

--#BLOCK_BEGIN# DropOutput


IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_B') AND sysstat & 0xf =
3) DROP TABLE Order_0_B
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_INC') AND sysstat & 0xf
= 3) DROP TABLE Order_0_INC


--#BLOCK_END# DropOutput

/*****************************************************************************/
--Create FC table in case force close was
```

```
-- not run
/*****************************************************************************/

--#BLOCK_BEGIN# MakeFC

DECLARE @fc_exists INT



SELECT @fc_exists = (
SELECT COUNT(1)

FROM sysobjects
WHERE
id = object_id('dbo.Order_0_FC') AND sysstat & 0xf = 3

)

IF (@fc_exists = 0)
EXEC('

SELECT
        *
INTO Order_0_FC
FROM
        Order_0_A
WHERE
        1=0
')



--#BLOCK_END# MakeFC

/*****************************************************************************/
-- Create the incremental table
/*****************************************************************************/

--#BLOCK_BEGIN# MakeINC


SELECT
        *
INTO Order_0_INC
FROM Order_0_TIN UNION ALL
SELECT * FROM Order_0_IL UNION ALL
SELECT * FROM Order_0_IR UNION ALL
SELECT * FROM Order_0_IRD UNION ALL
SELECT * FROM Order_0_IND UNION ALL
SELECT * FROM Order_0_IRM UNION ALL
SELECT * FROM Order_0_ILM UNION ALL
SELECT * FROM Order_0_FC UNION ALL
SELECT * FROM Order_0_IDM

--#BLOCK_END# MakeINC

/*****************************************************************************/
-- CR158: We want to load _IMI table and still keep the non-descending
-- order so that the clustered index on a fact table can be created
-- without sorting.  This way can speed up significantly in creating a
-- clustered index on a very large already sorted fact table.
/*****************************************************************************/

--#BLOCK_BEGIN# MakeIMI


SELECT
        *
INTO Order_0_IMI
FROM Order_0_A
WHERE date_key >= (SELECT MIN(date_key) FROM Order_0_INC)
```

```
UNION ALL
SELECT * FROM Order_0_INC
ORDER BY
       date_key
,      customerbillto_key
,      product_key
,      application_key
,      program_key
,      customershipto_key
,      territory_key
,      warehouse_key


--#BLOCK_END# MakeIMI

/*****************************************************************************/
-- Create the new fact table and incremental table
--
-- Note that transaction tables must be built before
-- these statements are run
/*****************************************************************************/

--#BLOCK_BEGIN# MakeNewFact


SELECT *
INTO Order_0_B
FROM Order_0_A s
WHERE s.date_key < (SELECT MIN(date_key) FROM Order_0_INC)
UNION ALL
SELECT * FROM Order_0_IMI

--#BLOCK_END# MakeNewFact

/*****************************************************************************/
-- Count processed, inserted rows
/*****************************************************************************/

--#BLOCK_BEGIN# SPResults

DECLARE @count_INC INT

BEGIN

SELECT @count_INC = (
SELECT COUNT(1)

FROM Order_0_INC
)

INSERT INTO adaptive_template_profile (token_name, number_rows)
SELECT 'PROCESSED', COUNT(1) FROM Order_0_MFL

INSERT INTO adaptive_template_profile (token_name, number_rows)
SELECT 'INSERTED', @count_INC - COUNT(1) FROM Order_0_TIN

END

--#BLOCK_END# SPResults

/*********************************************************************/
-- Set join order for SQL Server
/*********************************************************************/

--#BLOCK_BEGIN# ForcePlanOff

SET FORCEPLAN OFF

--#BLOCK_END# ForcePlanOff

/*****************************************************************************/
```

```
-- Drop temp tables and TXN and TIN table
/*****************************************************************************/

--#BLOCK_BEGIN# DropTempsAfter


IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_TIN') AND sysstat & 0xf
= 3) DROP TABLE Order_0_TIN
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_TMI') AND sysstat & 0xf
= 3) DROP TABLE Order_0_TMI
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_FC') AND sysstat & 0xf =
3) DROP TABLE Order_0_FC
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_TXN') AND sysstat & 0xf
= 3) DROP TABLE Order_0_TXN
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Concat_MFL') AND sysstat & 0xf =
3) DROP TABLE Concat_MFL
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_1ST') AND sysstat & 0xf
= 3) DROP TABLE Order_0_1ST
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_IL') AND sysstat & 0xf =
3) DROP TABLE Order_0_IL
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_IR') AND sysstat & 0xf =
3) DROP TABLE Order_0_IR
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_IRD') AND sysstat & 0xf
= 3) DROP TABLE Order_0_IRD
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_IND') AND sysstat & 0xf
= 3) DROP TABLE Order_0_IND
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_NFD') AND sysstat & 0xf
= 3) DROP TABLE Order_0_NFD
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_IRM') AND sysstat & 0xf
= 3) DROP TABLE Order_0_IRM
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_IDM') AND sysstat & 0xf
= 3) DROP TABLE Order_0_IDM
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_ILM') AND sysstat & 0xf
= 3) DROP TABLE Order_0_ILM
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_IMI') AND sysstat & 0xf
= 3) DROP TABLE Order_0_IMI


--#BLOCK_END# DropTempsAfter

--#TEMPLATE_END# load_state
 --#TEMPLATE_BEGIN# load_trans

/*****************************************************************************/
--
-- Copyright * 1997, Epiphany Marketing Software, Inc.  All Rights Reserved.
--
-- load_trans
--
-- Move transaction-like staging data into Fact table - create a temp
-- table with TXN extension that has all old rows along with new rows.
-- Also produce a TIN (TXN INC) table that has only the new rows
--
-- Note that the new table will also include all existing rows from
-- the Fact table.
--
/*****************************************************************************/

/*****************************************************************************/
-- Delete output tables
--
-- Output table is called TXN and includes old and new rows
--
-- Also, leave around _TIN as incremental table from this
-- procedure
--
-- We also create  a table called _TMI which contains all the
-- _TIN records plus the records of overlapping period from the
-- old existing fact table.
/*****************************************************************************/
```

```
--#BLOCK_BEGIN# RemoveOutput


IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_TXN') AND sysstat & 0xf
= 3) DROP TABLE Order_0_TXN
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_TMI') AND sysstat & 0xf
= 3) DROP TABLE Order_0_TMI
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_TIN') AND sysstat & 0xf
= 3) DROP TABLE Order_0_TIN


--#BLOCK_END# RemoveOutput

/*****************************************************************/
-- Set join order for SQL Server
/*****************************************************************/

--#BLOCK_BEGIN# ForcePlanOn

SET FORCEPLAN ON

--#BLOCK_END# ForcePlanOn

/*************************************************************************************/
-- Remove stuff already in  fact table
--
-- Note that currently this filter implies that once a transactional
-- fact entry is made it cannot be changed - and no further fact
-- entries on that date or any previous date can be made either
/*************************************************************************************/

--#BLOCK_BEGIN# CreateTIN


SELECT
        s.iss,
        s.ss_key,
        s.date_key,
        s.transtype_key,
        s.ikey seq
,       s.customerbillto_key
,       s.product_key
,       s.application_key
,       s.program_key
,       s.customershipto_key
,       s.territory_key
,       s.warehouse_key

,       s.net_price
 ,      s.number_units

INTO Order_0_TIN
FROM
        OrderStage_MAP s, bus_process b
WHERE
        NOT EXISTS (SELECT * FROM Order_0_A f WHERE
                s.iss = f.iss AND
                s.ss_key = f.ss_key AND
                f.date_key >= s.date_key)
AND     (
        (s.net_price <> 0)
 OR     (s.number_units <> 0)
        )
AND
        s.process_key = b.process_key AND b.process_name = 'LoadTrans'

--#BLOCK_END# CreateTIN

/*****************************************************************/
-- Set join order for SQL Server
/*****************************************************************/
```

```
--#BLOCK_BEGIN# ForcePlanOff

SET FORCEPLAN OFF

--#BLOCK_END# ForcePlanOff

/*****************************************************************************/
-- CR158: We want to load _TMI table and still keep the non-descending
-- order so that the clustered index on a fact table can be created
-- without sorting.  This way can speed up significantly in creating a
-- clustered index on a very large already sorted fact table.
/*****************************************************************************/

--#BLOCK_BEGIN# CreateTMI


SELECT
        *
INTO Order_0_TMI
FROM
        Order_0_A
WHERE
        date_key >= (SELECT MAX(date_key) FROM Order_0_TIN)
UNION ALL
SELECT
        *
FROM
        Order_0_TIN
ORDER BY
        date_key
,       customerbillto_key
,       product_key
,       application_key
,       program_key
,       customershipto_key
,       territory_key
,       warehouse_key


--#BLOCK_END# CreateTMI

/*****************************************************************************/
-- Insert everything into the new fact table
/*****************************************************************************/

--#BLOCK_BEGIN# CreateTXN


SELECT
        *
INTO Order_0_TXN
FROM
        Order_0_A s
WHERE s.date_key < (SELECT MAX(date_key) FROM Order_0_TIN)
UNION ALL
SELECT
        *
FROM
        Order_0_TMI f

--#BLOCK_END# CreateTXN

/*****************************************************************************/
-- Count inserted data and put results into communication table
/*****************************************************************************/

--#BLOCK_BEGIN# SPResults

BEGIN
```

```
INSERT INTO adaptive_template_profile (token_name, number_rows)
SELECT 'PROCESSED', COUNT(1) FROM OrderStage_MAP

INSERT INTO adaptive_template_profile (token_name, number_rows)
SELECT 'INSERTED', COUNT(1) FROM Order_0_TIN

END

--#BLOCK_END# SPResults

--#TEMPLATE_END# load_trans

 --#TEMPLATE_BEGIN# index_fact

/*****************************************************************************************/
--
-- Copyright * 1997, Epiphany Marketing Software, Inc.  All Rights Reserved.
--
-- Post processing after an extraction run
--
-- Reindex fact tables
-- CR158: added WITH SORTED_DATA in creating cluster index on fact table
--
-- Remove any temp tables generated during the extraction
--
/*****************************************************************************************/

/*****************************************************************************************/
-- Primary key index the fact table
/*****************************************************************************************/

--#BLOCK_BEGIN# PKIndexFact


EXEC('

CREATE UNIQUE INDEX XPKOrder_0_B ON Order_0_B
 (
  iss , ss_key , date_key , transtype_key , seq
 )
')


--#BLOCK_END# PKIndexFact

/*****************************************************************************************/
-- Inversion index the fact table
/*****************************************************************************************/

--#BLOCK_BEGIN# IEIndexFact


EXEC('

CREATE CLUSTERED INDEX XIEKOrder_0_B ON Order_0_B
 (
       date_key
,      customerbillto_key
,      product_key
,      application_key
,      program_key
,      customershipto_key
,      territory_key
,      warehouse_key
 ) WITH SORTED_DATA

')



--#BLOCK_END# IEIndexFact
```

```
/*****************************************************************************/
-- Remove any mapped tables
/*****************************************************************************/

--#BLOCK_BEGIN# RemoveTemps


IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.OrderStage_MAP') AND sysstat &
0xf = 3) DROP TABLE OrderStage_MAP


--#BLOCK_END# RemoveTemps

--#TEMPLATE_END# index_fact
 --#TEMPLATE_BEGIN# ren_trans

/*****************************************************************************/
--
-- Copyright * 1997, Epiphany Marketing Software, Inc.  All Rights Reserved.
--
-- ren_trans
--
-- Epiphany Marketing Software, 1997
--
-- Simply change the name of the transaction new table to the
-- actual fact table name - used for Fact tables that don't have
-- any stored procedure other than load_trans attached to them
--
/*****************************************************************************/

/*****************************************************************************/
-- Delete the output tables
/*****************************************************************************/

--#BLOCK_BEGIN# RemoveOutput


IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_B') AND sysstat & 0xf =
3) DROP TABLE Order_0_B
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_INC') AND sysstat & 0xf
= 3) DROP TABLE Order_0_INC


--#BLOCK_END# RemoveOutput

/*****************************************************************************/
-- Move all transaction rows into the correct new fact table
-- name. Note that we would use sp_rename, except it
-- doesn't work with DB name prefixes
--
-- TBD: Rename instead of re-select
/*****************************************************************************/

--#BLOCK_BEGIN# BuildNewFact


SELECT
        *
INTO Order_0_B
FROM
        Order_0_TXN

--#BLOCK_END# BuildNewFact

/*****************************************************************************/
-- Preserve incremental table
/*****************************************************************************/

--#BLOCK_BEGIN# BuildIncremental
```

```
SELECT
        *
INTO Order_0_INC
FROM
        Order_0_TIN

--#BLOCK_END# BuildIncremental

/*************************************************************************/
-- Count inserted data and put results into communication table
/*************************************************************************/

--#BLOCK_BEGIN# SPResults

BEGIN

INSERT INTO adaptive_template_profile (token_name, number_rows)
SELECT 'PROCESSED', COUNT(1) FROM Order_0_TXN

INSERT INTO adaptive_template_profile (token_name, number_rows)
SELECT 'INSERTED', COUNT(1) FROM Order_0_TXN

END

--#BLOCK_END# SPResults

/*************************************************************************/
-- Remove temp tables
/*************************************************************************/

--#BLOCK_BEGIN# RemoveTemps


IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_TXN') AND sysstat & 0xf
= 3) DROP TABLE Order_0_TXN
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_TIN') AND sysstat & 0xf
= 3) DROP TABLE Order_0_TIN
IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.Order_0_TMI') AND sysstat & 0xf
= 3) DROP TABLE Order_0_TMI


--#BLOCK_END# RemoveTemps

--#TEMPLATE_END# ren_trans

 --#TEMPLATE_BEGIN# map_keys

/*************************************************************************/
--
-- Copyright * 1997, Epiphany Marketing Software, Inc.  All Rights Reserved.
--
-- map_keys
--
-- Epiphany Marketing Software
--
-- Map dimension keys from Staging table and report
-- on unjoined rows
--
/*************************************************************************/

/*************************************************************************/
-- Remove output table
/*************************************************************************/

--#BLOCK_BEGIN# DropTemp


IF EXISTS (SELECT 1 FROM sysobjects WHERE id = object_id('dbo.OrderStage_MAP') AND sysstat &
0xf = 3) DROP TABLE OrderStage_MAP
```

```
--#BLOCK_END# DropTemp

/*********************************************************************/
-- Set join order for SQL Server
/*********************************************************************/

--#BLOCK_BEGIN# ForcePlanOn

SET FORCEPLAN ON

--#BLOCK_END# ForcePlanOn

/*********************************************************************/
-- Map dimension keys via Inner joins
/*********************************************************************/

--#BLOCK_BEGIN# MapAll


SELECT
        s.iss,
        s.ss_key,
        s.date_key,
        s.transtype_key,
        s.ikey,
        s.process_key

,       m_04.program_key program_key
,       m_03.application_key application_key
,       m_06.territory_key territory_key
,       m_02.product_key product_key
,       m_05.customer_key customershipto_key
,       m_01.customer_key customerbillto_key
,       m_07.warehouse_key warehouse_key


,       s.net_price
,       s.number_units

INTO OrderStage_MAP
FROM
        OrderStage s
,       ProgramMap_B m_04 (index = 1)
,       ApplicationMap_B m_03 (index = 1)
,       TerritoryMap_B m_06 (index = 1)
,       ProductMap_B m_02 (index = 1)
,       CustomerMap_B m_05 (index = 1)
,       CustomerMap_B m_01 (index = 1)
,       WarehouseMap_B m_07 (index = 1)

WHERE 1=1
AND     m_04.iss = s.iss AND m_04.program_sskey = s.program_sskey
AND     m_03.iss = s.iss AND m_03.application_sskey = s.application_sskey
AND     m_06.iss = s.iss AND m_06.territory_sskey = s.territory_sskey
AND     m_02.iss = s.iss AND m_02.product_sskey = s.product_sskey
AND     m_05.iss = s.iss AND m_05.customer_sskey = s.customershipto_sskey
AND     m_01.iss = s.iss AND m_01.customer_sskey = s.customerbillto_sskey
AND     m_07.iss = s.iss AND m_07.warehouse_sskey = s.warehouse_sskey

--#BLOCK_END# MapAll

/*********************************************************************/
-- Set join order for SQL Server
/*********************************************************************/

--#BLOCK_BEGIN# ForcePlanOff

SET FORCEPLAN OFF

--#BLOCK_END# ForcePlanOff
```

```
/*****************************************************************/
-- Look for unjoined data, Report on processed rows
/*****************************************************************/

--#BLOCK_BEGIN# SPResults


DECLARE @unjoined INT
DECLARE @processed INT

BEGIN

SELECT @processed = (
SELECT COUNT(1)

FROM OrderStage
)

SELECT @unjoined = (
SELECT @processed - COUNT(1)

FROM OrderStage_MAP
)

INSERT INTO adaptive_template_profile (token_name, number_rows)
SELECT 'UNJOINED', @unjoined

INSERT INTO adaptive_template_profile (token_name, number_rows)
SELECT 'PROCESSED', @processed

INSERT INTO adaptive_template_profile (token_name, number_rows)
SELECT 'INSERTED', @processed - @unjoined

END

--#BLOCK_END# SPResults

/*****************************************************************/
-- Index this temp table
/*****************************************************************/

--#BLOCK_BEGIN# IndexMap


EXEC('
CREATE INDEX XOrderStage_MAP ON OrderStage_MAP
(
 iss, ss_key, date_key, ikey
)
')


--#BLOCK_END# IndexMap

--#TEMPLATE_END# map_keys
 --#TEMPLATE_BEGIN# upd_unj

/*****************************************************************/
--
-- Copyright * 1997, Epiphany Marketing Software, Inc.  All Rights Reserved
--
-- upd_unj
--
-- Epiphany Marketing Software
--
-- Update all dimension keys to 'UNKNOWN' in staging table
-- where referential integrity fails
--
/*****************************************************************/
```

```
/*********************************************************************/
-- Count the number of rows to update in the staging table - that is, those
-- that have at least one Foreign key where referential integrity fails
/*********************************************************************/

--#BLOCK_BEGIN# CountUnj

BEGIN

INSERT INTO adaptive_template_profile (token_name, number_rows)
SELECT 'PROCESSED', COUNT(1) FROM OrderStage

INSERT INTO adaptive_template_profile (token_name, number_rows)
SELECT 'MODIFIED', COUNT(1)
FROM
      OrderStage s
WHERE 1=0
OR NOT EXISTS (SELECT 1 FROM ProgramMap_B m_04 WHERE m_04.iss = s.iss AND m_04.program_sskey =
program_sskey)
OR NOT EXISTS (SELECT 1 FROM ApplicationMap_B m_03 WHERE m_03.iss = s.iss AND
m_03.application_sskey = application_sskey)
OR NOT EXISTS (SELECT 1 FROM TerritoryMap_B m_06 WHERE m_06.iss = s.iss AND
m_06.territory_sskey = territory_sskey)
OR NOT EXISTS (SELECT 1 FROM ProductMap_B m_02 WHERE m_02.iss = s.iss AND m_02.product_sskey =
product_sskey)
OR NOT EXISTS (SELECT 1 FROM CustomerMap_B m_05 WHERE m_05.iss = s.iss AND m_05.customer_sskey
= customershipto_sskey)
OR NOT EXISTS (SELECT 1 FROM CustomerMap_B m_01 WHERE m_01.iss = s.iss AND m_01.customer_sskey
= customerbillto_sskey)
OR NOT EXISTS (SELECT 1 FROM WarehouseMap_B m_07 WHERE m_07.iss = s.iss AND
m_07.warehouse_sskey = warehouse_sskey)

END

--#BLOCK_END# CountUnj

/*********************************************************************/
-- Update foreign keys where referential integrity fails
/*********************************************************************/

--#BLOCK_BEGIN# UpdateUnjprogram_sskey

UPDATE OrderStage SET program_sskey = 'UNKNOWN'
WHERE NOT EXISTS (SELECT 1 FROM ProgramMap_B m
WHERE m.iss = OrderStage.iss AND m.program_sskey = OrderStage.program_sskey)

--#BLOCK_END# UpdateUnjprogram_sskey

--#BLOCK_BEGIN# UpdateUnjapplication_sskey

UPDATE OrderStage SET application_sskey = 'UNKNOWN'
WHERE NOT EXISTS (SELECT 1 FROM ApplicationMap_B m
WHERE m.iss = OrderStage.iss AND m.application_sskey = OrderStage.application_sskey)

--#BLOCK_END# UpdateUnjapplication_sskey

--#BLOCK_BEGIN# UpdateUnjterritory_sskey

UPDATE OrderStage SET territory_sskey = 'UNKNOWN'
WHERE NOT EXISTS (SELECT 1 FROM TerritoryMap_B m
WHERE m.iss = OrderStage.iss AND m.territory_sskey = OrderStage.territory_sskey)

--#BLOCK_END# UpdateUnjterritory_sskey

--#BLOCK_BEGIN# UpdateUnjproduct_sskey

UPDATE OrderStage SET product_sskey = 'UNKNOWN'
WHERE NOT EXISTS (SELECT 1 FROM ProductMap_B m
WHERE m.iss = OrderStage.iss AND m.product_sskey = OrderStage.product_sskey)

--#BLOCK_END# UpdateUnjproduct_sskey
```

```
--#BLOCK_BEGIN# UpdateUnjcustomershipto_sskey

UPDATE OrderStage SET customershipto_sskey = 'UNKNOWN'
WHERE NOT EXISTS (SELECT 1 FROM CustomerMap_B m
WHERE m.iss = OrderStage.iss AND m.customer_sskey = OrderStage.customershipto_sskey)

--#BLOCK_END# UpdateUnjcustomershipto_sskey



--#BLOCK_BEGIN# UpdateUnjcustomerbillto_sskey

UPDATE OrderStage SET customerbillto_sskey = 'UNKNOWN'
WHERE NOT EXISTS (SELECT 1 FROM CustomerMap_B m
WHERE m.iss = OrderStage.iss AND m.customer_sskey = OrderStage.customerbillto_sskey)

--#BLOCK_END# UpdateUnjcustomerbillto_sskey

--#BLOCK_BEGIN# UpdateUnjwarehouse_sskey

UPDATE OrderStage SET warehouse_sskey = 'UNKNOWN'
WHERE NOT EXISTS (SELECT 1 FROM WarehouseMap_B m
WHERE m.iss = OrderStage.iss AND m.warehouse_sskey = OrderStage.warehouse_sskey)

--#BLOCK_END# UpdateUnjwarehouse_sskey



--#TEMPLATE_END# upd_unj
```

Note, additional semantic types and adaptive templates can be imported into the system 100.